Go Build Web

github.com/mijia

About Me

- Backend Engineer: Go, Python, Scala
- Frontend Engineer: JavaScript, HTML, CSS, React
- Platform Engineer: PaaS, Docker, Cluster Orchestration
- Data Engineer: Python, Go, Spark, Thrift
- github.com/mijia

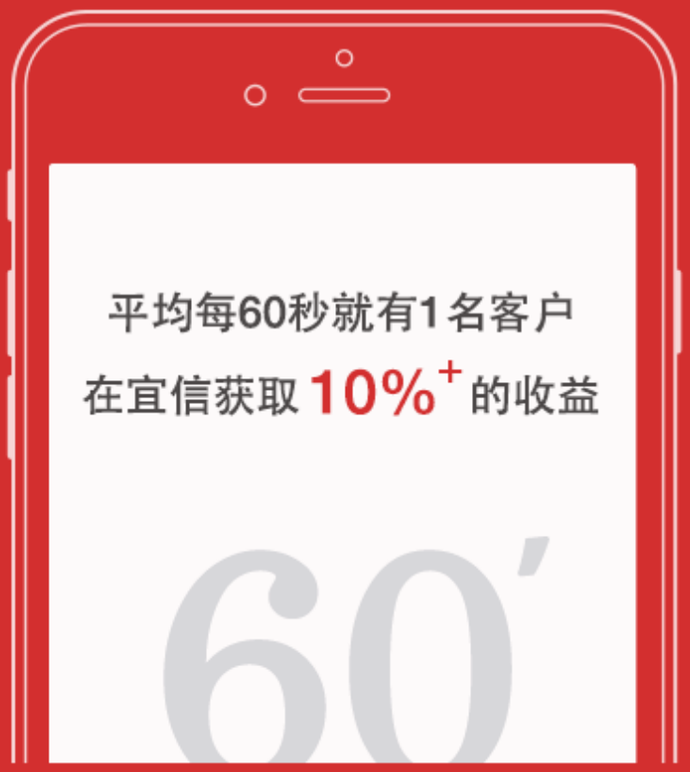
宜信 CreditEase | 指旺理财 好收益有指旺

免费注册 · 立即登录

指旺理财 — 最省心的理财应用

指旺理财依托于宜信强大的金融实力，通过互联网化的科学设计，
提供更安全、更高收益和更灵活的手机理财服务。

注册立领新手大礼包



Web Experiences

- ~ 2007: J2EE with Struts, Spring MVC, Hibernate
- 2008 - 2012: Python Django, Tornado, Flask
- 2013 - 2014: Scala on Play
- 2014 - Present: Go …

Build Web App

- Talk about building web applications*
- Functional Web Server/App Server
- Frameworks vs Packages
- Developer Tools

* Not only just api web servers, frontend stuff involved
* Information may be outdated

Hello World

```
func hello(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, “Hello, 世界”)
}

func main() {
    http.Handle(“/hello”, hello)
    log.Fatal(http.ListenAndServe(":8080", nil))
}

// Standard library so good!
// That easy ?!
```

Requests

Middleware

Routing

Handler

Handler

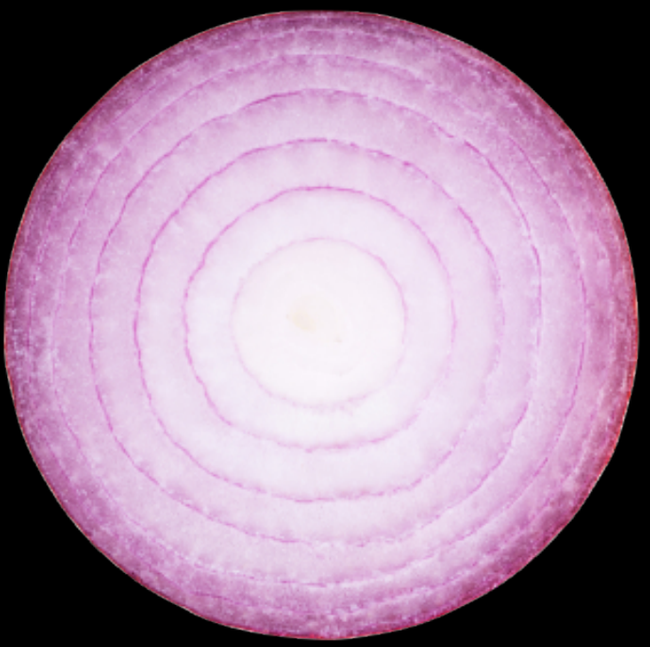
Handler

Handler

Handler

Responses

Like an Onion



Inside the handler

```
def handler([context], request, response):
    // parse form or query from request

    // get some service object from context
    // e.g. cache, db, session, user auth

    // we use some model objects or POJOs
    // user = models.User(name="john")
    // CRUD the model object, biz logic

// then render the result into response
// html, json, xml and etc…
// or error/panic
```

Basically
we do stuff
like this

Beyond Hello World

- Stdlib: Request, Response, Server, Protocol, Handler
- Routing/Mux, Middleware
- Context
- Model/ORM
- Render/Template: HTML, JSON, XML
- Errors, Session, Caching, Form Validation, Security, Log, Config, Encryption, Restful, Graceful Shutdown, Queue, Jobs …

Go Web Frameworks

- beego.me
- gin-gonic.github.io/gin/
- github.com/go-martini/martini
- github.com/zenazn/goji
- revel.github.io
- ……

	net/http	Beego	Gin	Martini	Goji	Revel
Routing	DefaultMux, string max-length matching	Param, Regex, Namespace, Tree Search	Param, Group, HttpRouter (Prefix Tree)	Param, Group, Handler* (any callable), Linear	DefaultMux, Param, Regex, Pattern Iface, State Machine	Param, pathtree.Tree
Middleware	No	Filters	Middleware, per-group, per-route	Context/Injector	Middleware	Filters, Interceptors, per-action
Context	No x/net/context	beego/context input, output	gin.Context (Req, Res, Metadata Facade)	Context/Injector	web.C Params, Envs	No
Model/ORM	No	beego/orm MySQL, Postgres, Sqlite3	No	No	No	No
Render	html/template text/template	html/template layout support	html/template, JSON, XML	No	No	html/template, app/views/ *.html
Advanced	/debug/…	Lots of utils, session, log, Cache, deploy…	Recovery, Log, Sentry, Binding	*No Longer Maintained, too magical	Minimalistic	Hot Reload, Session, Cache, Debug, WS, I18N

Go Web Packages

- julienschmidt/httprouter
  - dozens of frameworks based on this pkg
- codegangsta/negroni
  - Middlewares: dozens of negroni compatible middleware pkgs on github.com
- www.gorillatoolkit.org
  - context, mux, reverse, cookie, session, websocket
- ……

Go Proverbs

- Simple, Poetic, Pithy
    - The bigger the interface, the weaker the abstraction.
- We don’t like all the magic since we are coming from Scala/Play, :(
- Composing, Interface, Adaptor

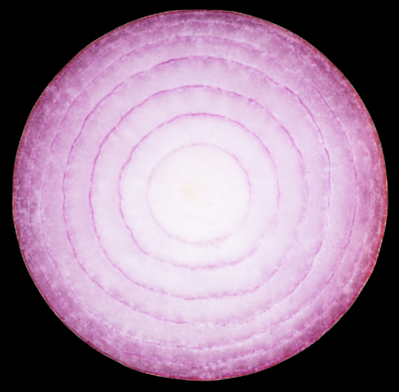
Compose and Adapt

Compose Into Big Handler

```
// our factory adaptor
type Handler func(
    ctx context.Context,
    w http.ResponseWriter,
    r *http.Request) context.Context
```

```
// httprouter/router.go
type Handle func(http.ResponseWriter, *http.Request, Params)
```

```
// net/http/server.go
type Handler interface {
    ServeHTTP(ResponseWriter, *Request)
}
```



Middleware

Wrap middlewares and router into the “onion”, compose into big net/http handler

Go Proverbs

A little copying is better than a little dependency.

```
// Our factory adaptor
type Middleware interface {
      ServeHTTP(ctx context.Context, w http.ResponseWriter, r *http.Request,
                        next Handler) context.Context
}
```

```
// Negroni Middleware
type Handler interface {
  ServeHTTP(rw http.ResponseWriter, r *http.Request, next http.HandlerFunc)
}
```

We Love Go Pkgs

- So we select the best but simple packages to adapt and compose into our own web toolkit, which provides routing, context, middleware, render, form params and validation …… Ain't that much magic

- Will love go pkgs more if Go team solved the vendor/dependency problem, :)

- github.com/mijia/sweb

Developer Tools

- ORMs (why is this here?)
- Frontend Assets (nodejs, npm, webpack…)
- Hot Reload (edit, build, refresh)
- Distribution Packges (make dist)

ORMs / Models

- Most ORMs for Go using reflection
  - Proverb: “Clear is better than clever.”
  - Proverb: “Reflection is never clear.”
- go generate: generate go code according to database scheme (Scala Slick)
- github.com/mijia/modelq: it’s a tool
- No joins, sql computations, but we can interface with “jmoiron/sqlx”

```
//go:generate modelq -db=root@/xxx -pkg=models -driver=mysql
    -schema=xxx -tables=user -template=models/modelq.tmpl
```

```
▼+User : struct
    [fields]
   +CreatedAt : time.Time
   +Email : string
   +Id : int64
   +Name : string
   +Password : string
   +Role : int
   +RoleReferenceId : int64
   +Salt : string
   +Status : int
   +UpdatedAt : time.Time
    [methods]
   +Delete(dbtx gmq.DbTx) : int64, error
   +Get(dbtx gmq.DbTx) : User, error
   +Insert(dbtx gmq.DbTx) : User, error
   +String() : string
   +Update(dbtx gmq.DbTx) : int64, error
```

```
▼-_UserQuery : struct
    [embedded]
   +gmq.Query : gmq.Query
    [methods]
   +GroupBy(by ) : _UserQuery
   +Iterate(dbtx gmq.DbTx, functor UserRowVisitor) : error
   +Limit(offsets ) : _UserQuery
   +List(dbtx gmq.DbTx) : []User, error
   +One(dbtx gmq.DbTx) : User, error
   +OrderBy(by ) : _UserQuery
   +Page(number, size int) : _UserQuery
   +Run(dbtx gmq.DbTx) : sql.Result, error
   +Where(f gmq.Filter) : _UserQuery
```

```
▼-_UserObjs : struct
    [fields]
   -fcMap : map[string]string
    [methods]
   +ColumnCreatedAt(p ) : gmq.Column
   +ColumnEmail(p ) : gmq.Column
   +ColumnId(p ) : gmq.Column
   +ColumnName(p ) : gmq.Column
   +ColumnPassword(p ) : gmq.Column
   +ColumnRole(p ) : gmq.Column
   +ColumnRoleReferenceId(p ) : gmq.Column
   +ColumnSalt(p ) : gmq.Column
   +ColumnStatus(p ) : gmq.Column
   +ColumnUpdatedAt(p ) : gmq.Column
   +Delete() : _UserQuery
   +FilterCreatedAt(op string, p time.Time, ps ) : gmq.Filter
   +FilterEmail(op string, p string, ps ) : gmq.Filter
   +FilterId(op string, p int64, ps ) : gmq.Filter
   +FilterName(op string, p string, ps ) : gmq.Filter
   +FilterPassword(op string, p string, ps ) : gmq.Filter
   +FilterRole(op string, p int, ps ) : gmq.Filter
   +FilterRoleReferenceId(op string, p int64, ps ) : gmq.Filter
   +FilterSalt(op string, p string, ps ) : gmq.Filter
   +FilterStatus(op string, p int, ps ) : gmq.Filter
   +FilterUpdatedAt(op string, p time.Time, ps ) : gmq.Filter
   +Insert(obj User) : _UserQuery
   +Names() : string, string, string
   +Select(fields ) : _UserQuery
   +Update(obj User, fields ) : _UserQuery
   -columns(fields ) : []gmq.Column
   -columnsWithData(obj User, fields ) : []gmq.Column
   -newFilter(name, op string, params ) : gmq.Filter
   -toUser(columns []gmq.Column, rb []sql.RawBytes) : User
```

Frontend Assets

- JavaScripts, Stylesheets, Images
  - JavaScript compiler/transpiler and bundler: babel, browserify, CoffeeScript
  - CSS: we also have stylus which can be compiled into css
  - Images: sprite image (Go Image Lib) and generate sprite@{1,2,3}x.styl files as stylus variables and mixins
- All assets needs fingerprint for browser cache expires
  - Go Template: {{ assets “images/desktop/logo.png” }}
- Why not gulp, grant, webpack … some real frontend toolsets?

Build a tool

- github.com/mijia/gobuildweb
  - Assets management: JavaScript, CSS, Images
  - Hot reload for rebuild -> kill -> run, also hot reload for templates
  - Make distribution package and cross compile
  - Don’t recommend to use
- Shift to Webpack + Makefile: let pros to worry about those staff
  - @fswatch $(GO_FILES) $(TEMPLATES) | xargs -n1 -I{} make restart || make kill
  - https://github.com/mijia/web-starter-kit

What we learned

- Go Standard Library is so good.
- 他山之石，可以攻玉
- Love simple, love compose, love interface, love Go Pkgs
- No desire of magic
- Love tools, very easy to make cli tools using Go, to boost productivity
- Use python to generate Go code, :)
  - Equals{Int,Int8,Int16,Int32,Int64,String…}Slice

Thank you