Go Library / Framework Roundup

Library

- Long term maintenance
- Composable
- Flexible
- Find your own
- "Your own Practices"

Framework

- Prototyping
- Lock-in (usually)
- Batteries included
- Kitchen sink included
- "Best Practices"

Libraries

net/http

- Robust and battle tested
- To get started:
    - https://go.dev/doc/articles/wiki/

gorillatoolkit.org

gorilla/mux: is a powerful URL router and dispatcher

gorilla/reverse: produces reversible regular expressions for regexp-muxes

gorilla/rpc: implements RPC over HTTP with codec for JSON-RPC

gorilla/schema: converts form values to a struct

gorilla/securecookie: encodes and decodes secure cookie values

gorilla/sessions: saves cookie and filesystem sessions

gorilla/websocket: implements the WebSocket protocol

gorilla/csrf: provides Cross Site Request Forgery (CSRF) prevention

gorilla/handlers: is a collection of useful net/http handlers

Authentication

- golang.org/x/oauth2
- github.com/markbates/goth
- *Your favourite Cloud Provider*

The new kid on the block:

- github.com/duo-labs/webauthn

Permissions

- github.com/casbin/casbin/v3
- biscuitsec.org
- openpolicyagent.org
- *Your favourite Cloud Provider*

Database

- github.com/jackc/pgx/v4
- github.com/mattn/go-sqlite3
- github.com/go-sql-driver/mysql

Avoid:

- github.com/lib/pq
- MongoDB

Database Migration

- github.com/golang-migrate/migrate
- github.com/pressly/goose
- Or write a for-loop:
  - e.g. github.com/joncalhoun/migrate

Templating

- github.com/google/safehtml/template
  - same as **html/template**, but with extra protection
  - contextual escaping
  - security model **https://pkg.go.dev/html/template#hdr-Security_Model**

Note, for dev, auto-reload templates:

- *https://github.com/loov/watchrun/blob/master/examples/watchjs-templates/server.go#L15*

Logging

Structured:

- github.com/rs/zerolog
- github.com/go-kit/log
- go.uber.org/zap

Unstructured:

- log

However, for all of them avoid global logger and instead pass as dependencies.

Testing

- github.com/matryer/is
    - github.com/zeebo/assert (or write your own)
- github.com/rogpeppe/go-internal/testscript
- github.com/ory/dockertest
- github.com/go-rod/rod

CLI

- flag
- github.com/zeebo/clingy

Popular:

- github.com/spf13/cobra
- github.com/urfave/cli

Observability

- OpenTelemetry (go.opentelemetry.io/otel)

OpenTracing and OpenCensus are both deprecated.

gokit.io

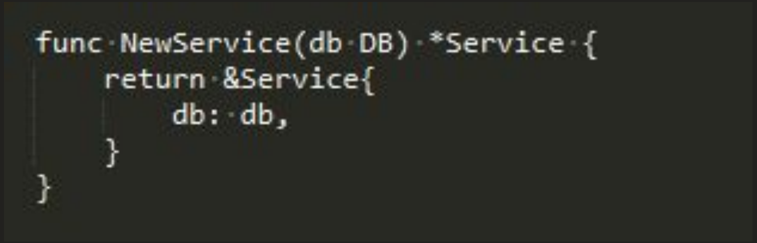
- Common things needed for (micro-)services:
  - auth
  - circuitbreaker, ratelimit
  - metrics, tracing
  - service discovery
  - transport

Security

- golang.org/x/crypto/nacl
- filippo.io/age

Dependency Injection

- It's usually easier and clearer to wire things up manually:



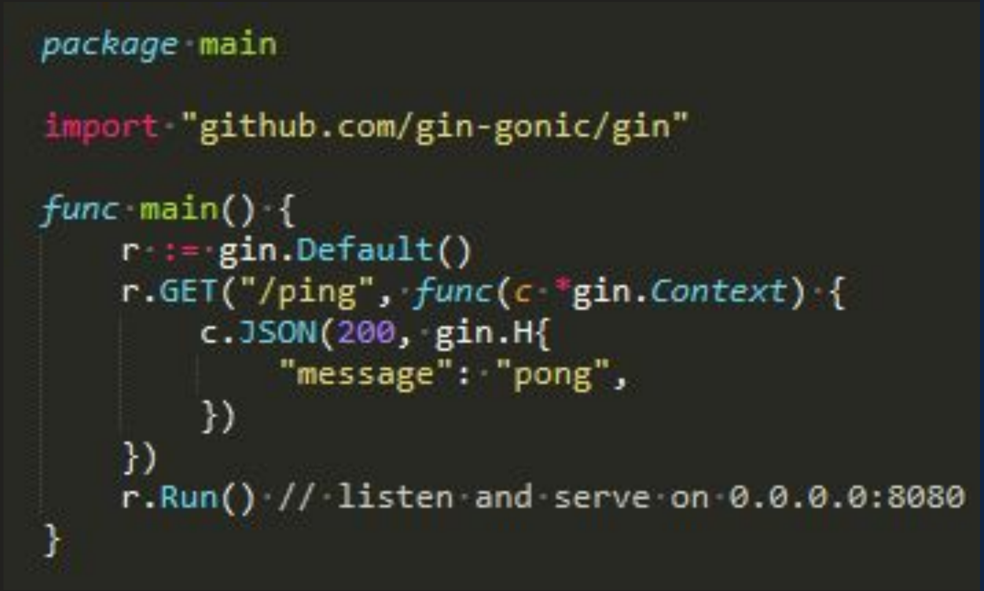
```go
func NewService(db DB) *Service {
    return &Service{
        db: db,
    }
}
```

However, if you really must:

- *github.com/google/wire*

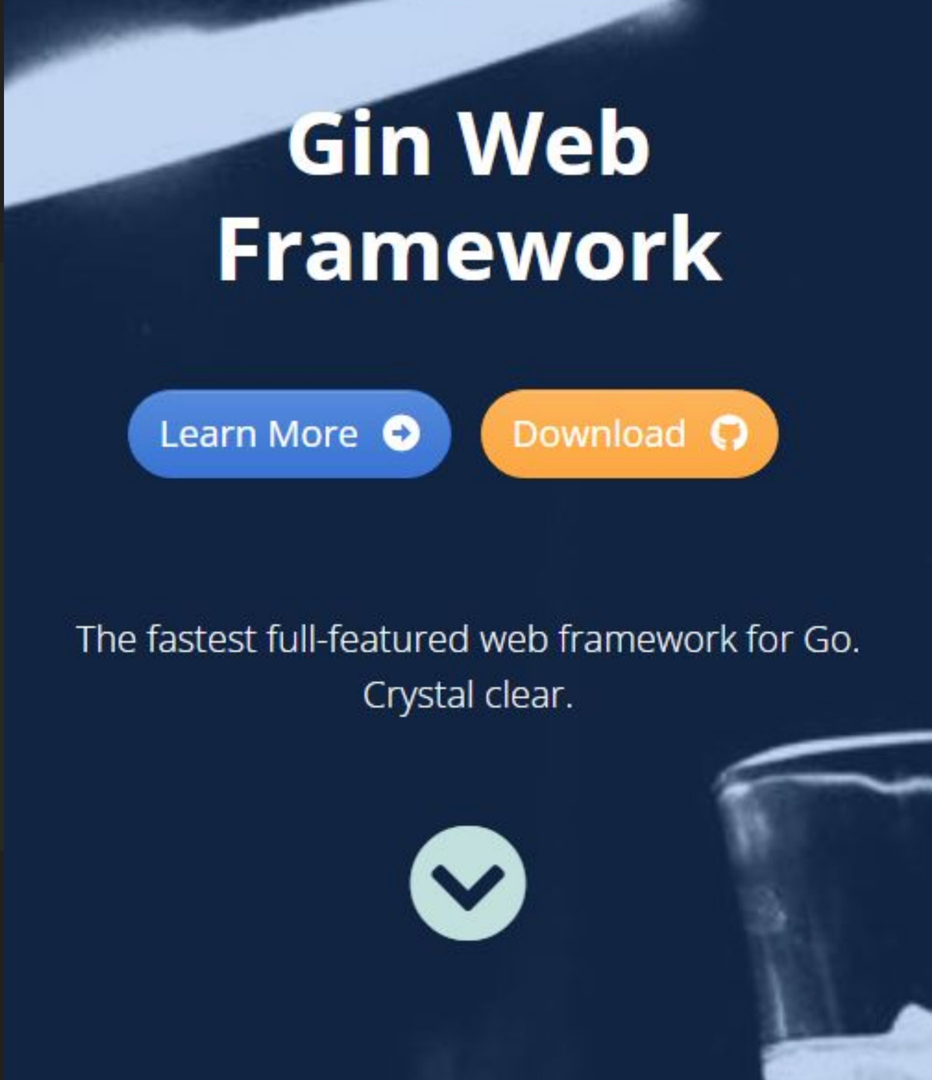
Frameworks

gin-gonic.com



```go
package main

import "github.com/gin-gonic/gin"

func main() {
	r := gin.Default()
	r.GET("/ping", func(c *gin.Context) {
		c.JSON(200, gin.H{
			"message": "pong",
		})
	})
	r.Run() // listen and serve on 0.0.0.0:8080
}
```

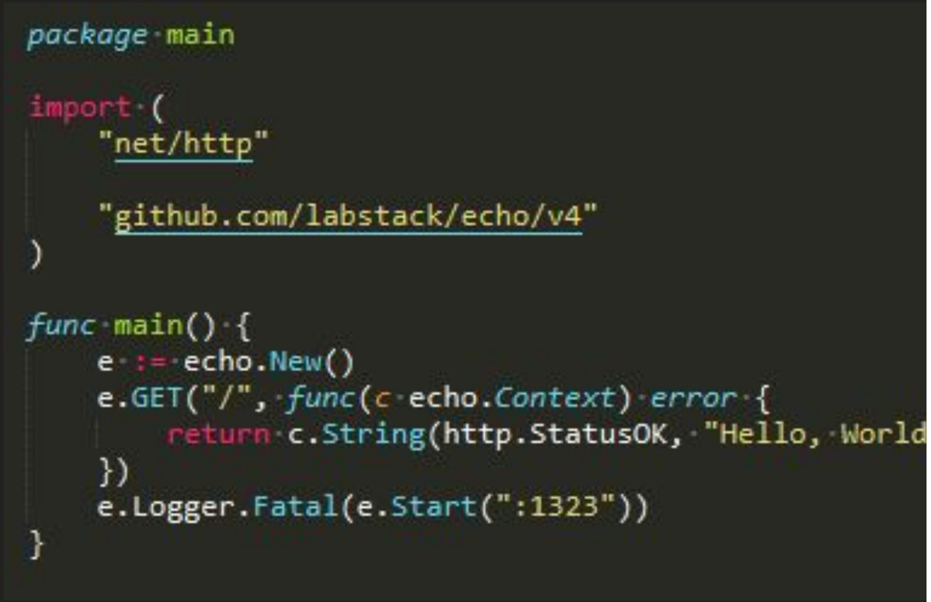
Mostly a convenience wrapper and middleware around net/http.



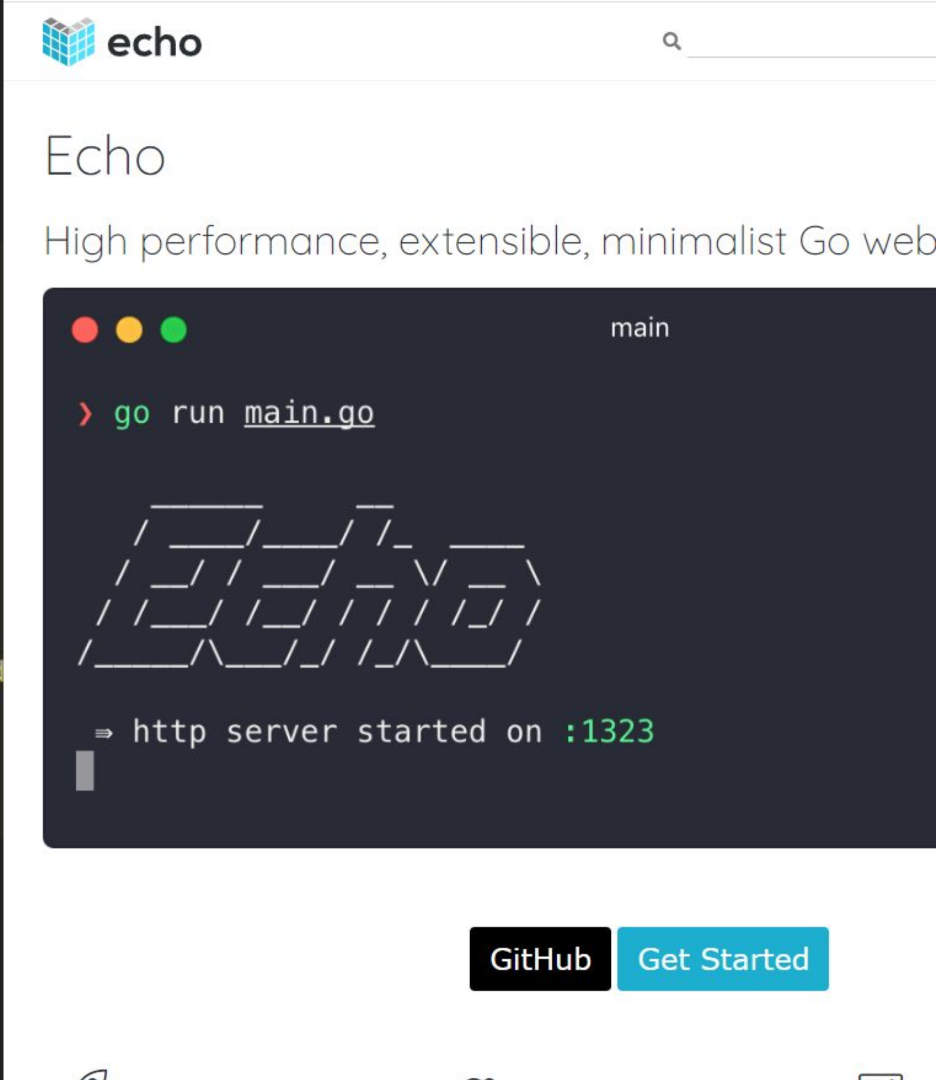
Gin Web Framework

The fastest full-featured web framework for Go. Crystal clear.

echo.labstack.com



```go
package main

import (
	"net/http"

	"github.com/labstack/echo/v4"
)

func main() {
	e := echo.New()
	e.GET("/", func(c echo.Context) error {
		return c.String(http.StatusOK, "Hello, World
	})
	e.Logger.Fatal(e.Start(":1323"))
}
```



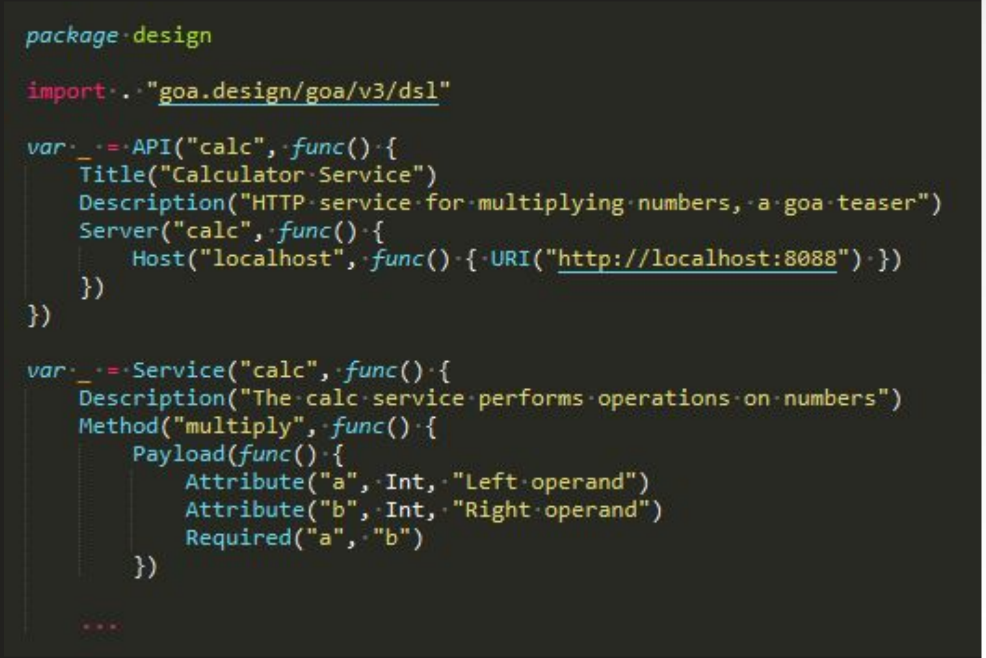
Echo

High performance, extensible, minimalist Go web

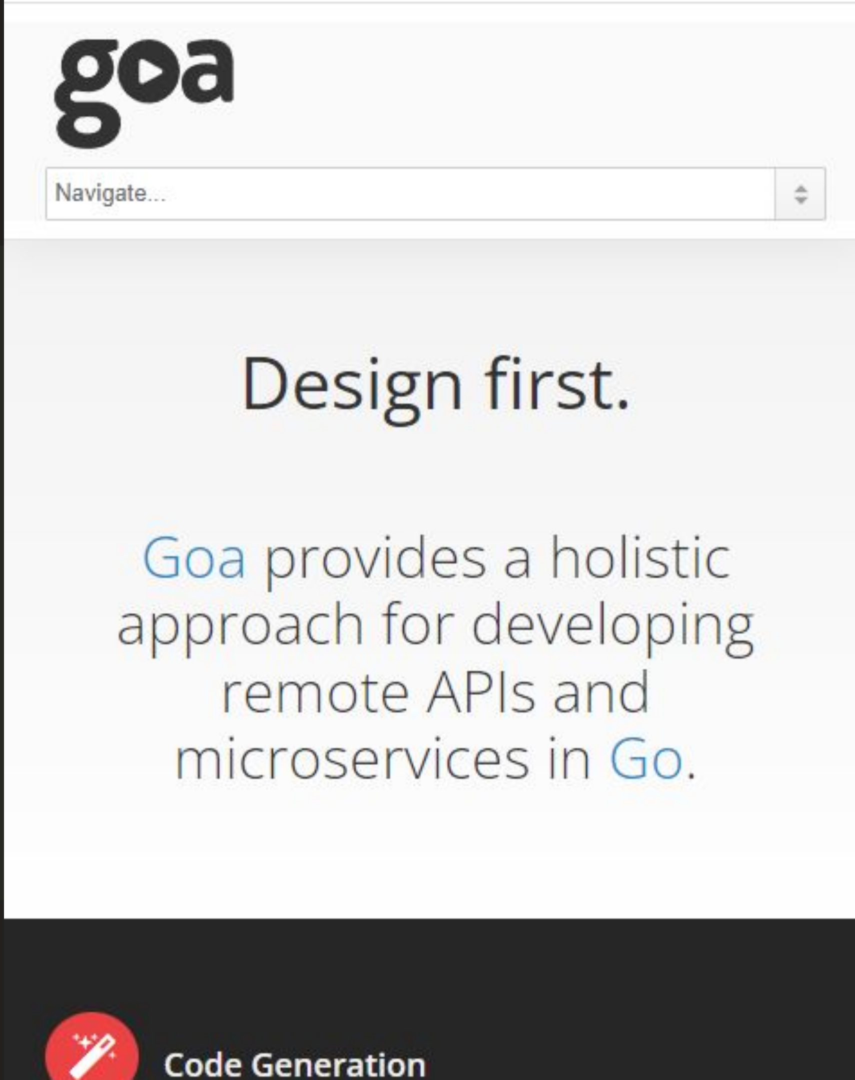
```
❯ go run main.go

   ____    __
  / __/___/ /  ___
 / _// __/ _ \/ _ \
/___/\__/_//_/\___/

 ⇒ http server started on :1323
```

Similar to gin, but better organized middleware and documentation.

goa.design



```go
package design

import . "goa.design/goa/v3/dsl"

var _ = API("calc", func() {
    Title("Calculator Service")
    Description("HTTP service for multiplying numbers, a goa teaser")
    Server("calc", func() {
        Host("localhost", func() { URI("http://localhost:8088") })
    })
})

var _ = Service("calc", func() {
    Description("The calc service performs operations on numbers")
    Method("multiply", func() {
        Payload(func() {
            Attribute("a", Int, "Left operand")
            Attribute("b", Int, "Right operand")
            Required("a", "b")
        })

    ...
```

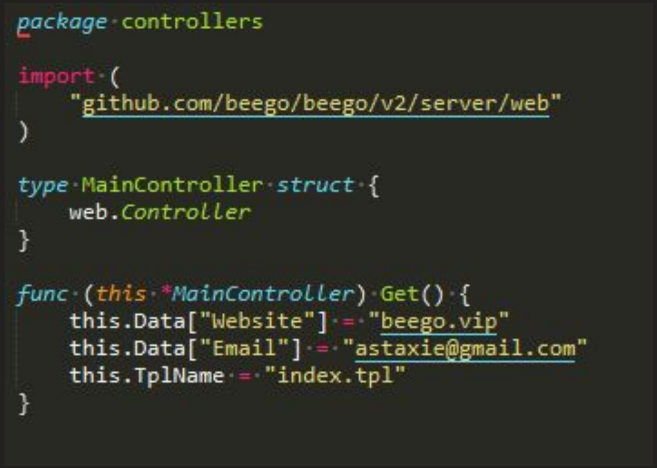


Design first.

Goa provides a holistic approach for developing remote APIs and microservices in Go.

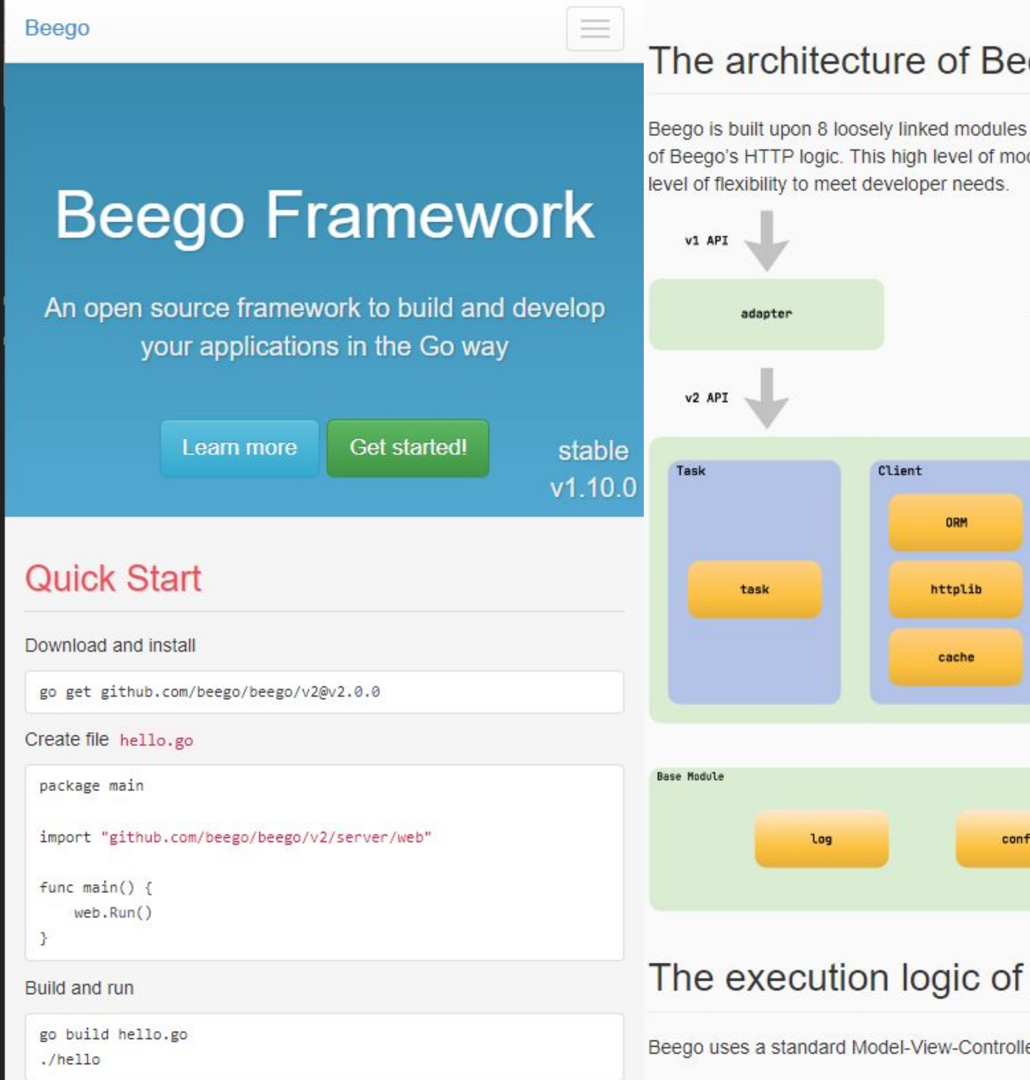
A DSL for generating web server boilerplate.

beego.vip



```go
package controllers

import (
    "github.com/beego/beego/v2/server/web"
)

type MainController struct {
    web.Controller
}

func (this *MainController) Get() {
    this.Data["Website"] = "beego.vip"
    this.Data["Email"] = "astaxie@gmail.com"
    this.TplName = "index.tpl"
}
```

Following in the steps of many PHP MVC frameworks.



Beego

Beego Framework

An open source framework to build and develop your applications in the Go way

stable v1.10.0

Quick Start

Download and install

```
go get github.com/beego/beego/v2@v2.0.0
```

Create file `hello.go`

```go
package main

import "github.com/beego/beego/v2/server/web"

func main() {
    web.Run()
}
```

Build and run

```
go build hello.go
./hello
```

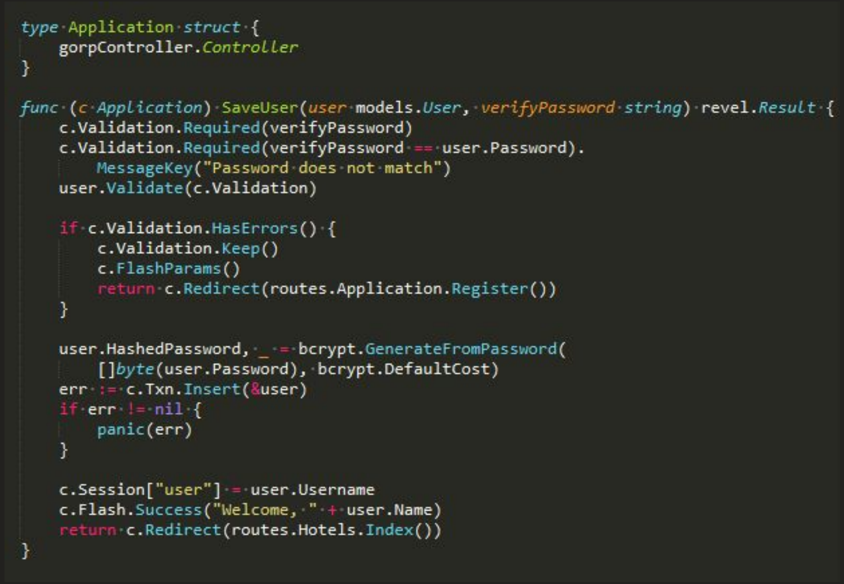
The architecture of Be

Beego is built upon 8 loosely linked modules
of Beego's HTTP logic. This high level of mo
level of flexibility to meet developer needs.

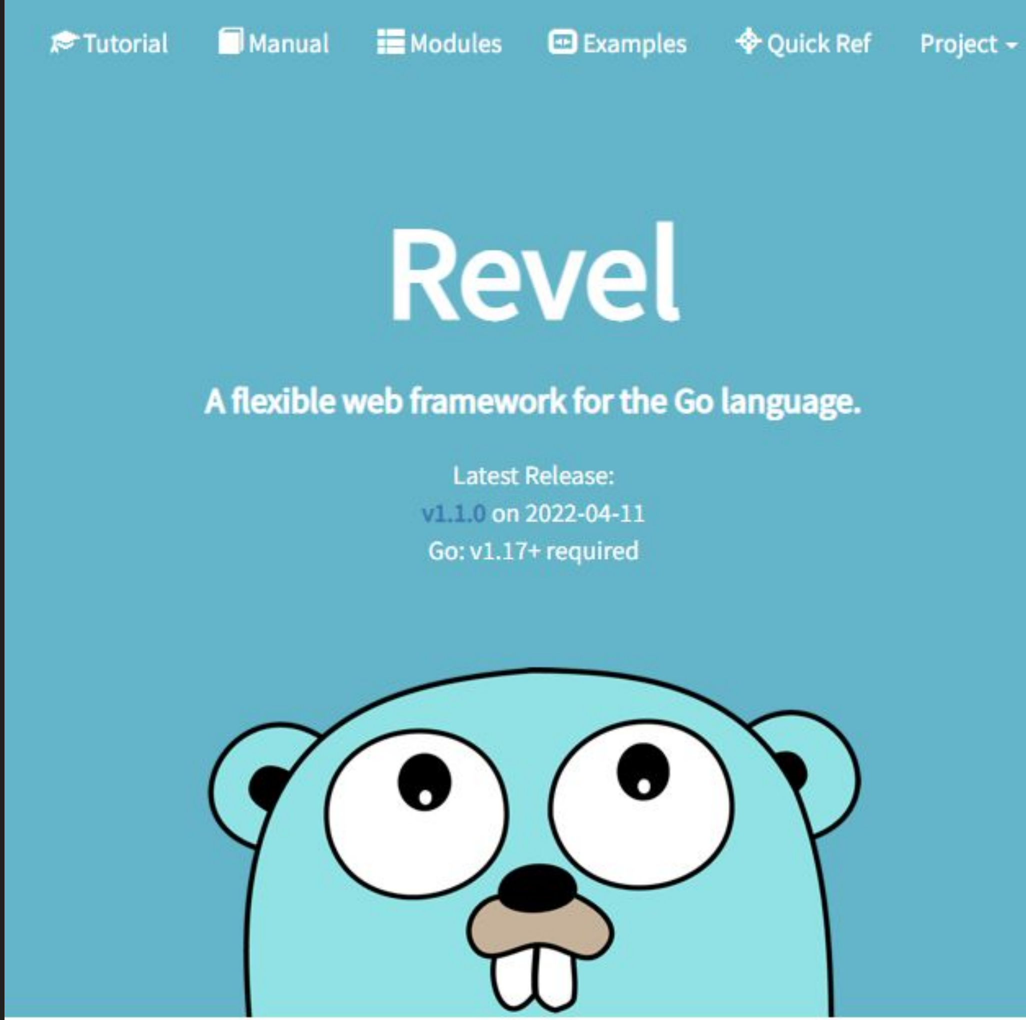
The execution logic of

Beego uses a standard Model-View-Controlle

revel.github.io/



```go
type Application struct {
    gorpController.Controller
}

func (c Application) SaveUser(user models.User, verifyPassword string) revel.Result {
    c.Validation.Required(verifyPassword)
    c.Validation.Required(verifyPassword == user.Password).
        MessageKey("Password does not match")
    user.Validate(c.Validation)

    if c.Validation.HasErrors() {
        c.Validation.Keep()
        c.FlashParams()
        return c.Redirect(routes.Application.Register())
    }

    user.HashedPassword, _ = bcrypt.GenerateFromPassword(
        []byte(user.Password), bcrypt.DefaultCost)
    err := c.Txn.Insert(&user)
    if err != nil {
        panic(err)
    }

    c.Session["user"] = user.Username
    c.Flash.Success("Welcome, " + user.Name)
    return c.Redirect(routes.Hotels.Index())
}
```



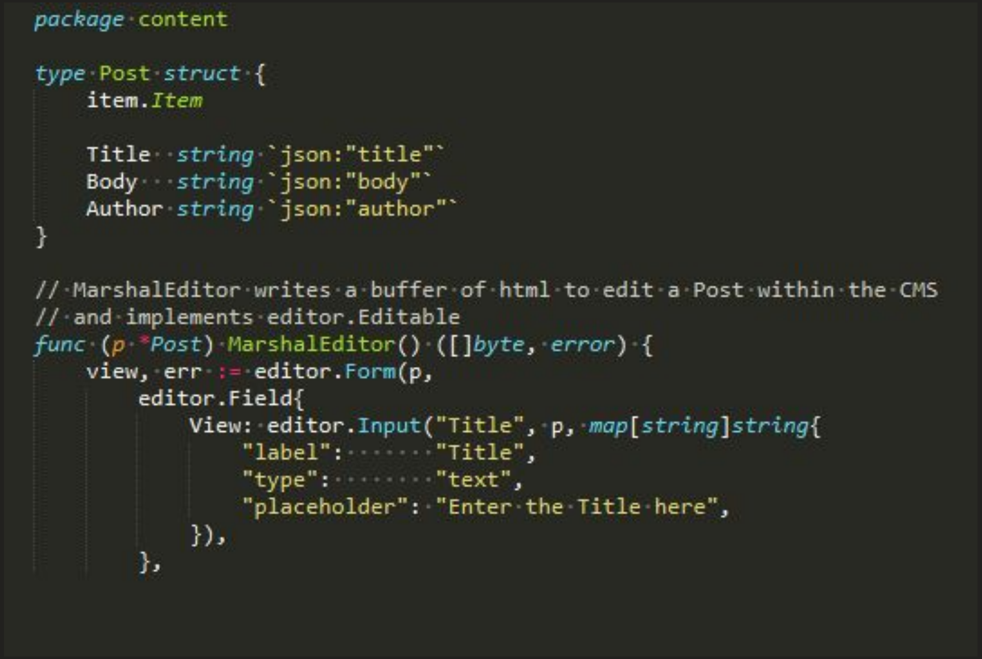
Revel

A flexible web framework for the Go language.

Latest Release:
v1.1.0 on 2022-04-11
Go: v1.17+ required

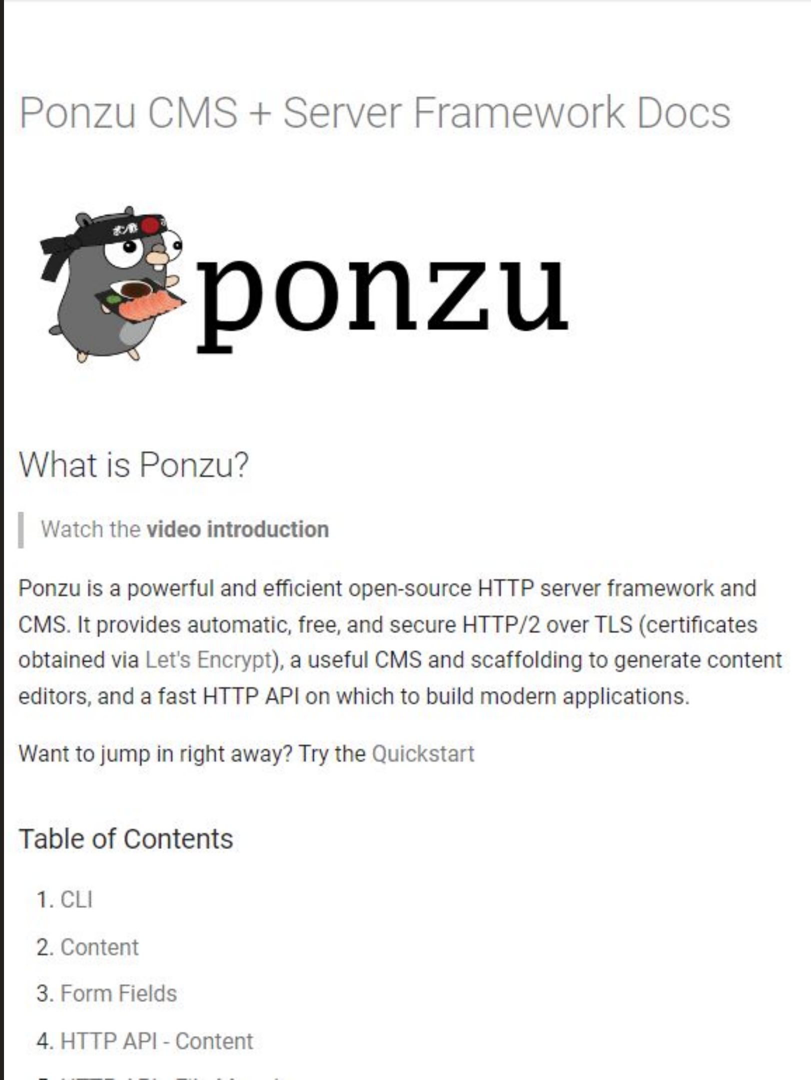
MVC PHP style,
and excellent examples

docs.ponzu-cms.org/



```go
package content

type Post struct {
    item.Item

    Title  string `json:"title"`
    Body   string `json:"body"`
    Author string `json:"author"`
}

// MarshalEditor writes a buffer of html to edit a Post within the CMS
// and implements editor.Editable
func (p *Post) MarshalEditor() ([]byte, error) {
    view, err := editor.Form(p,
        editor.Field{
            View: editor.Input("Title", p, map[string]string{
                "label":       "Title",
                "type":        "text",
                "placeholder": "Enter the Title here",
            }),
        },
```

Nice CMS, however,
no longer maintained



Ponzu CMS + Server Framework Docs

ponzu

What is Ponzu?

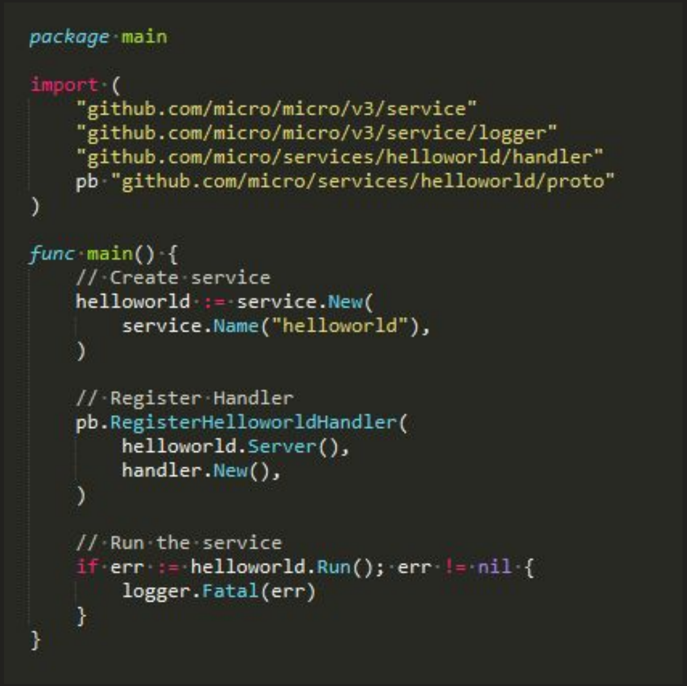
> Watch the **video introduction**

Ponzu is a powerful and efficient open-source HTTP server framework and CMS. It provides automatic, free, and secure HTTP/2 over TLS (certificates obtained via Let's Encrypt), a useful CMS and scaffolding to generate content editors, and a fast HTTP API on which to build modern applications.

Want to jump in right away? Try the Quickstart

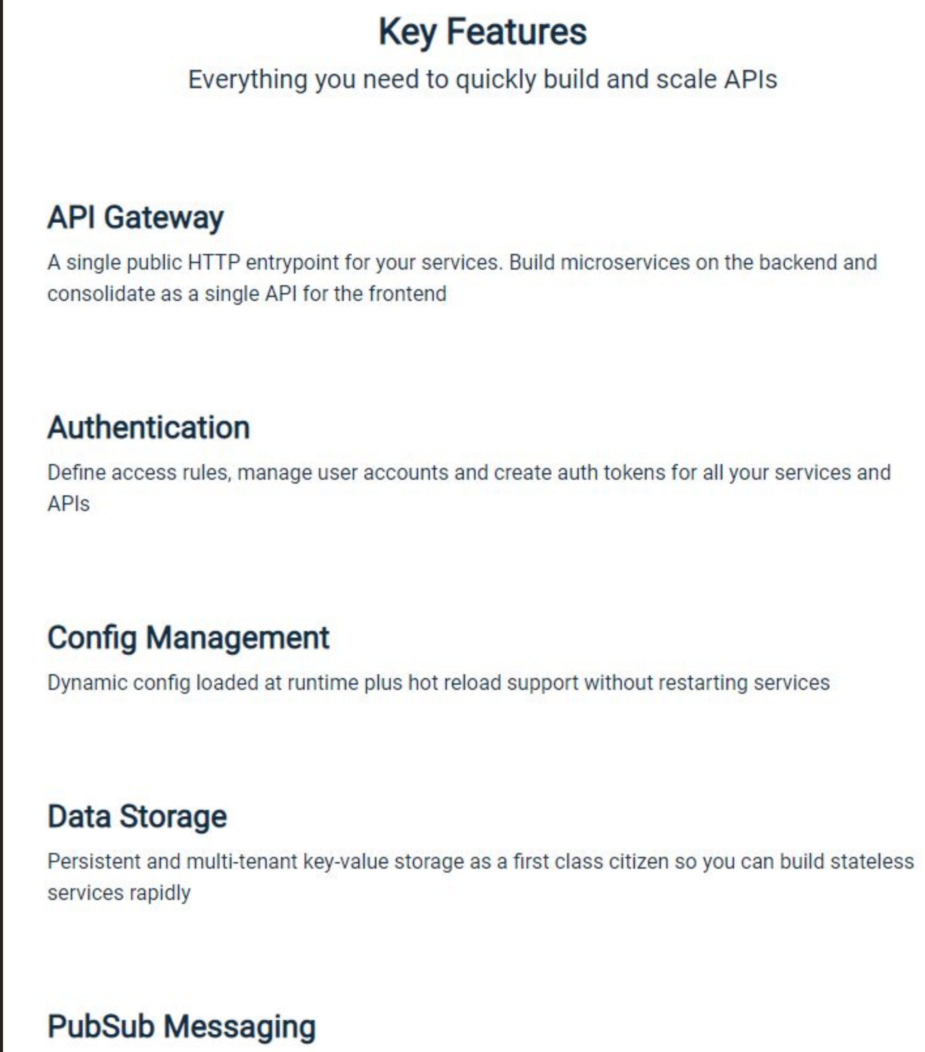
Table of Contents

github.com/micro/micro



```go
package main

import (
    "github.com/micro/micro/v3/service"
    "github.com/micro/micro/v3/service/logger"
    "github.com/micro/services/helloworld/handler"
    pb "github.com/micro/services/helloworld/proto"
)

func main() {
    // Create service
    helloworld := service.New(
        service.Name("helloworld"),
    )

    // Register Handler
    pb.RegisterHelloworldHandler(
        helloworld.Server(),
        handler.New(),
    )

    // Run the service
    if err := helloworld.Run(); err != nil {
        logger.Fatal(err)
    }
}
```

Microservice framework



Key Features

Everything you need to quickly build and scale APIs

API Gateway

A single public HTTP entrypoint for your services. Build microservices on the backend and consolidate as a single API for the frontend

Authentication

Define access rules, manage user accounts and create auth tokens for all your services and APIs

Config Management

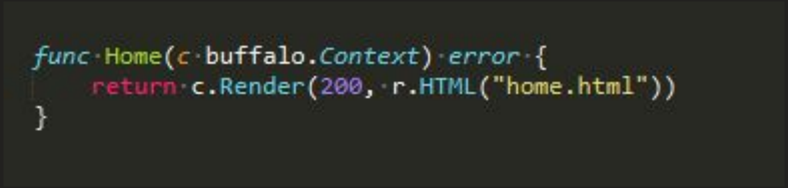
Dynamic config loaded at runtime plus hot reload support without restarting services

Data Storage

Persistent and multi-tenant key-value storage as a first class citizen so you can build stateless services rapidly

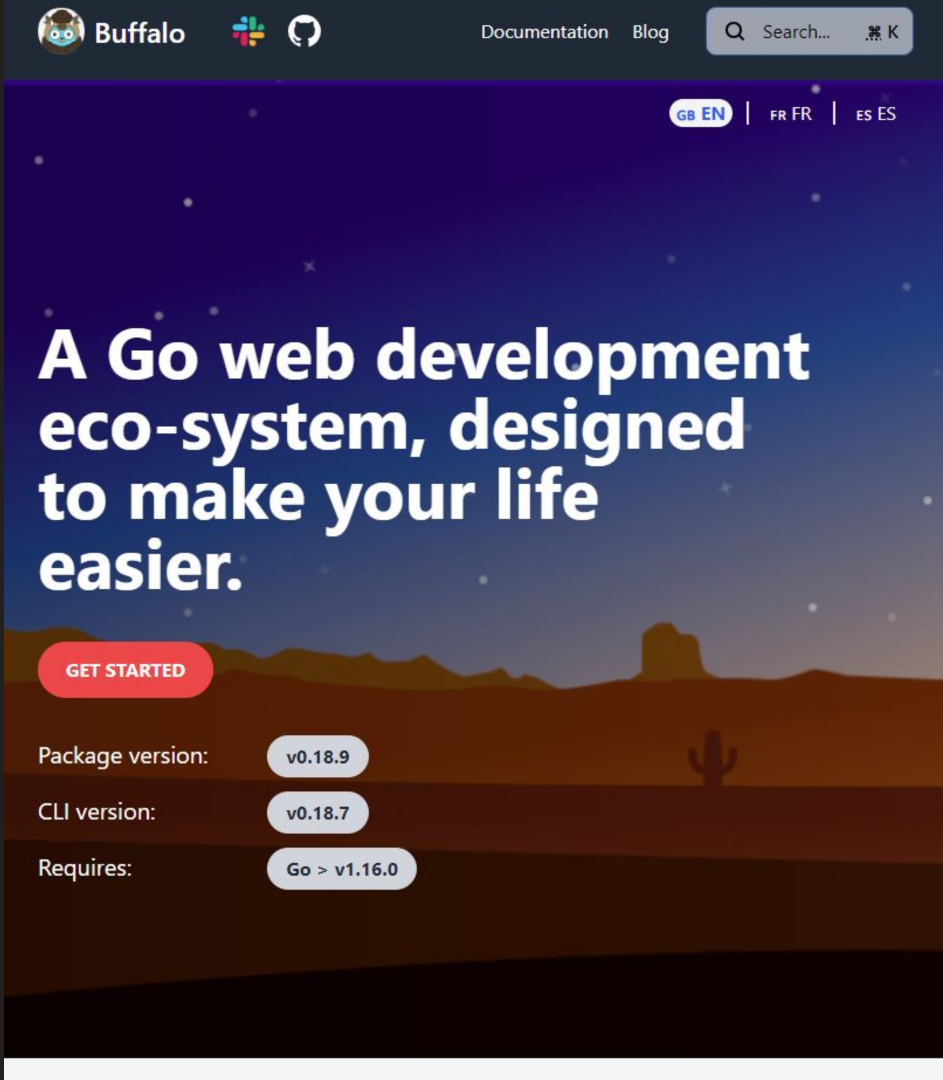
PubSub Messaging

gobuffalo.io



```go
func Home(c buffalo.Context) error {
    return c.Render(200, r.HTML("home.html"))
}
```

Rails like experience.



A Go web development eco-system, designed to make your life easier.

Package version: v0.18.9

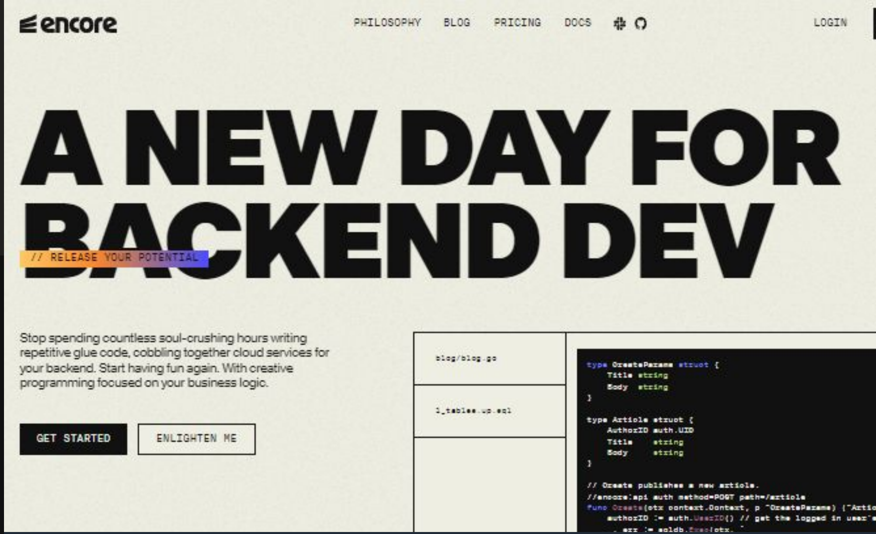
CLI version: v0.18.7

Requires: Go > v1.16.0

encore.dev

```
type Article struct {
    AuthorID auth.UID
    Title    string
    Body     string
}

// Create publishes a new article.
//encore:api auth method=POST path=/article
func Create(ctx context.Context, p *CreateParams) (*Article, error) {
    authorID := auth.UserID() // get the logged in user's ID
    _, err := sqldb.Exec(ctx, `
        INSERT INTO articles (author, title, body) VALUES ($1, $2, $3)
    `, authorID, p.Title, p.Body)
    if err != nil {
        return nil, err
    }
```

Custom Go compiler that wires things together.

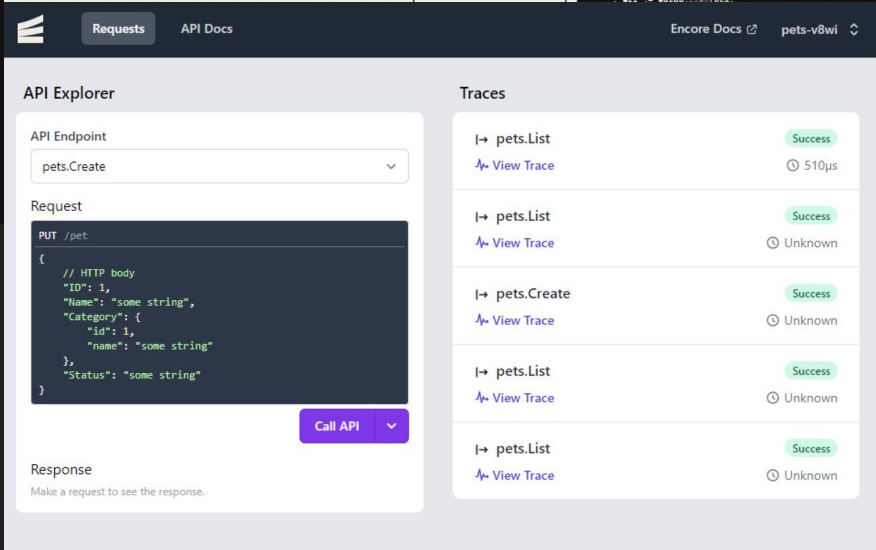


Stop spending countless soul-crushing hours writing repetitive glue code, cobbling together cloud services for your backend. Start having fun again. With creative programming focused on your business logic.

blog/blog.go

1_tables.up.sql

```
type CreateParams struct {
    Title string
    Body  string
}

type Article struct {
    AuthorID auth.UID
    Title    string
    Body     string
}

// Create publishes a new article.
//encore:api auth method=POST path=/article
func Create(ctx context.Context, p *CreateParams) (*Artic
    authorID := auth.UserID() // get the logged in user's
```



API Explorer

API Endpoint: pets.Create

Request

```
PUT /pet
{
    // HTTP body
    "ID": 1,
    "Name": "some string",
    "Category": {
        "id": 1,
        "name": "some string"
    },
    "Status": "some string"
}
```

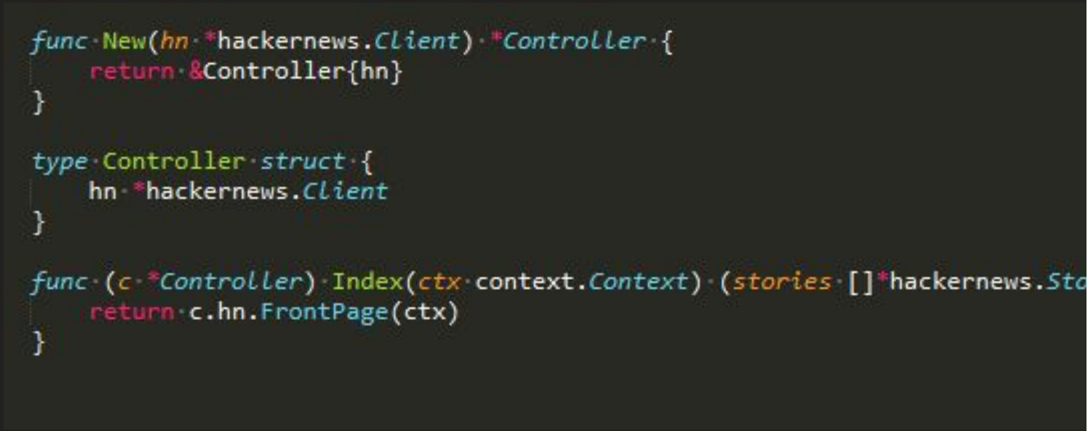
Response

Make a request to see the response.

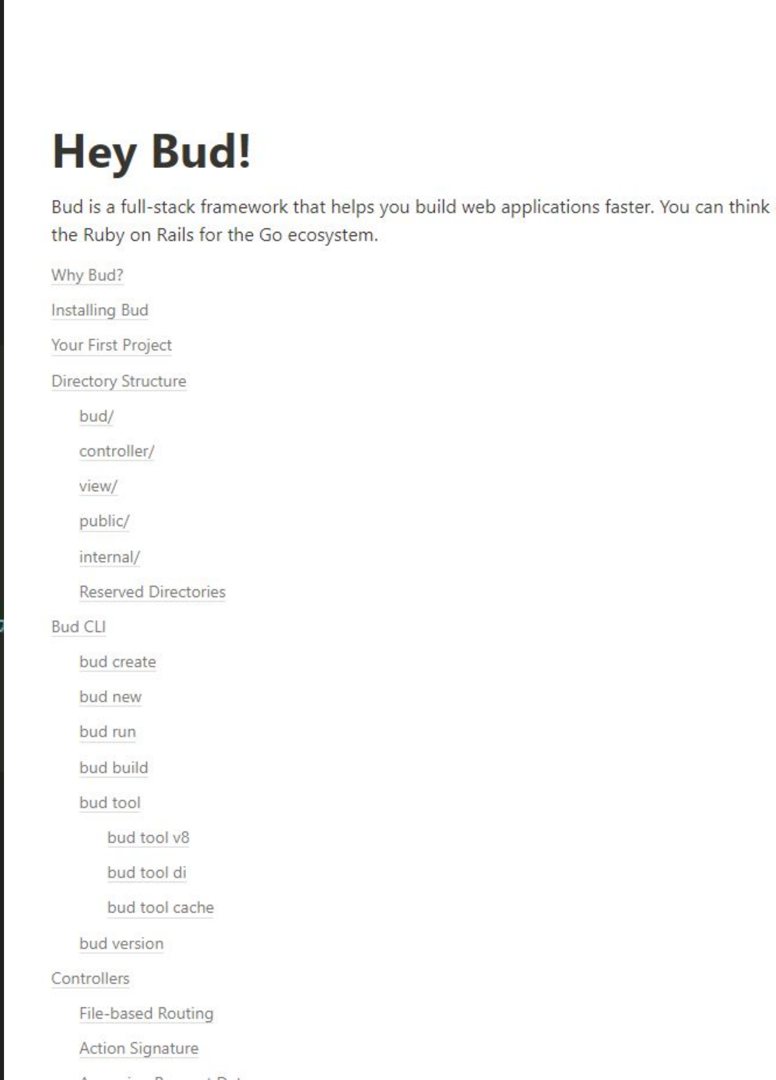
Traces

pets.List — View Trace	Success — 510µs
pets.List — View Trace	Success — Unknown
pets.Create — View Trace	Success — Unknown
pets.List — View Trace	Success — Unknown
pets.List — View Trace	Success — Unknown

github.com/livebud/bud



```go
func New(hn *hackernews.Client) *Controller {
    return &Controller{hn}
}

type Controller struct {
    hn *hackernews.Client
}

func (c *Controller) Index(ctx context.Context) (stories []*hackernews.Sto
    return c.hn.FrontPage(ctx)
}
```



Hey Bud!

Bud is a full-stack framework that helps you build web applications faster. You can think the Ruby on Rails for the Go ecosystem.

Still in development, but looks cool.

net/http

do you even need a framework?

egonelbre.com/server-and-a-database

Thanks for listening