

Создание вебсайта на Golang

11 ноября 2016

Владимир Ананьев
АО "Программный Регион"

Зачем создавать сайты на Golang?

- Всегда нужны графические интерфейсы для различных подсистем
- Очень просто
- Быстро
- Без дополнительных настроек **web-серверов** (типа IIS для ASP.NET, Apache для PHP и т.п.)

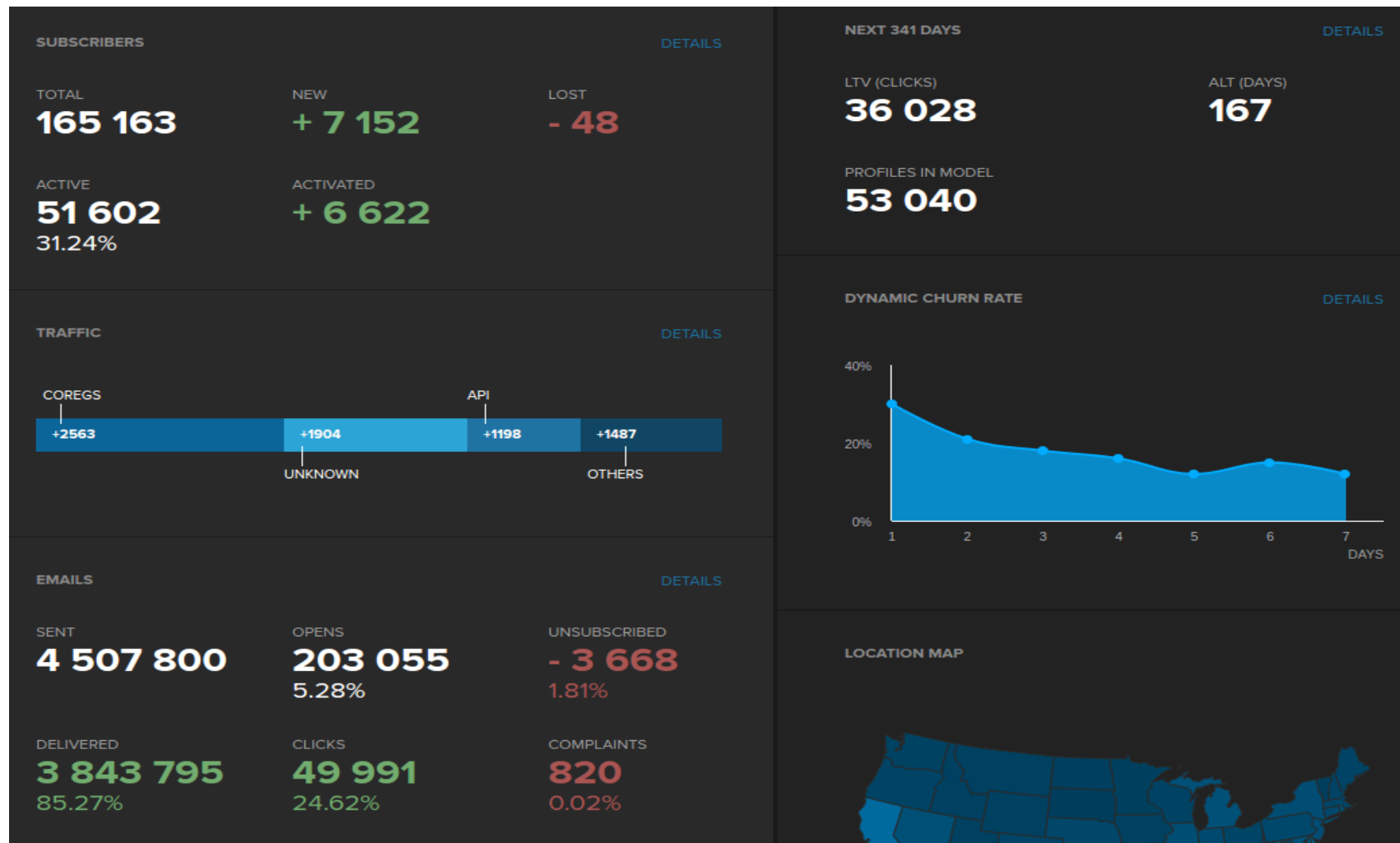
Примечание: лучше всегда сразу делать графический веб-интерфейс, а не платформо-зависимый системный интерфейс (окна и т.п.)

- Но есть и свои недостатки при использовании Golang сайтов

Как мы используем Golang сайты в компании



Управление аналитическими системами



- martini framework + custom design

Системы мониторинга

Rorschach	Alerts	Rules	Senders
<div><div><div><div>● EventStatCollector</div><div><div>ID: 58231b2a3f98cd00011d601c</div><div>Active since: Wed, 09 Nov 2016 12:48:42</div><div>Type: prometheus</div><div>Rule: EventStatCollector</div><div>Host: HZ53</div><div>App: greenplum</div><div>Message: production instance of greenplum has changed heartbeat id.</div></div></div><div><input checked="" type="checkbox"/> </div></div></div>			
<div><div><div><div>● EventStatCollector</div><div></div></div><div><input checked="" type="checkbox"/> </div></div></div>			
<div><div><div><div>● ConsulExporter</div><div></div></div><div><input checked="" type="checkbox"/> </div></div></div>			
<div><div><div><div>● ConsulExporter</div><div></div></div><div></div></div></div>			
<div><div><div><div>● ConsulExporter</div><div></div></div><div></div></div></div>			
<div><div><div><div>● ConsulExporter</div><div></div></div><div></div></div></div>			

- Самописный фреймворк на основе net/http + Bootstrap


Примечание: самописный фреймворк - это буквально 2 функции :)

Системы конфигурации

The screenshot displays the Consul web interface. At the top, there is a navigation bar with a logo on the left and several tabs: SERVICES, NODES, KEY/VALUE (which is highlighted with a purple border), ACL, and DC1 (highlighted with a green border). A settings gear icon is located on the far right of the navigation bar. Below the navigation bar, the main content area is divided into two panels. The left panel, titled with a '/' and a '+' icon, contains a list of configuration keys: Activation, ActivationLastRunDate, ActivationSyncConfig, Activator, Actuator, AerospikeConfig, AlarmistConfig, Alerts, Anton, AutoReporterConfig, Beta, and BrainConfig. The right panel is titled 'Create Key' and contains a text input field with a '/' character. Below the input field, there is a note: 'To create a folder, end the key with /'. A large text area for the value is located below the input field, with a note: 'Value can be any format and length'. At the bottom of the right panel, there is a 'CREATE' button.

*consul configuration & discovery (www.consul.io (<https://www.consul.io>))

Системы передачи сообщений

 NSQ Streams Nodes Counter Lookup Documentation GitHub v0.3.8

Streams / Tracking

Topic: Tracking

Empty Queue Delete Topic Pause Topic

Topic Message Queue

NSQd Host	Depth	Memory + Disk	Messages	Channels
✗ HZ39:4151 (136.243.39.70:4151)	0	0 + 0	93,831,276	4
Total:	0	0 + 0	93,831,276	4

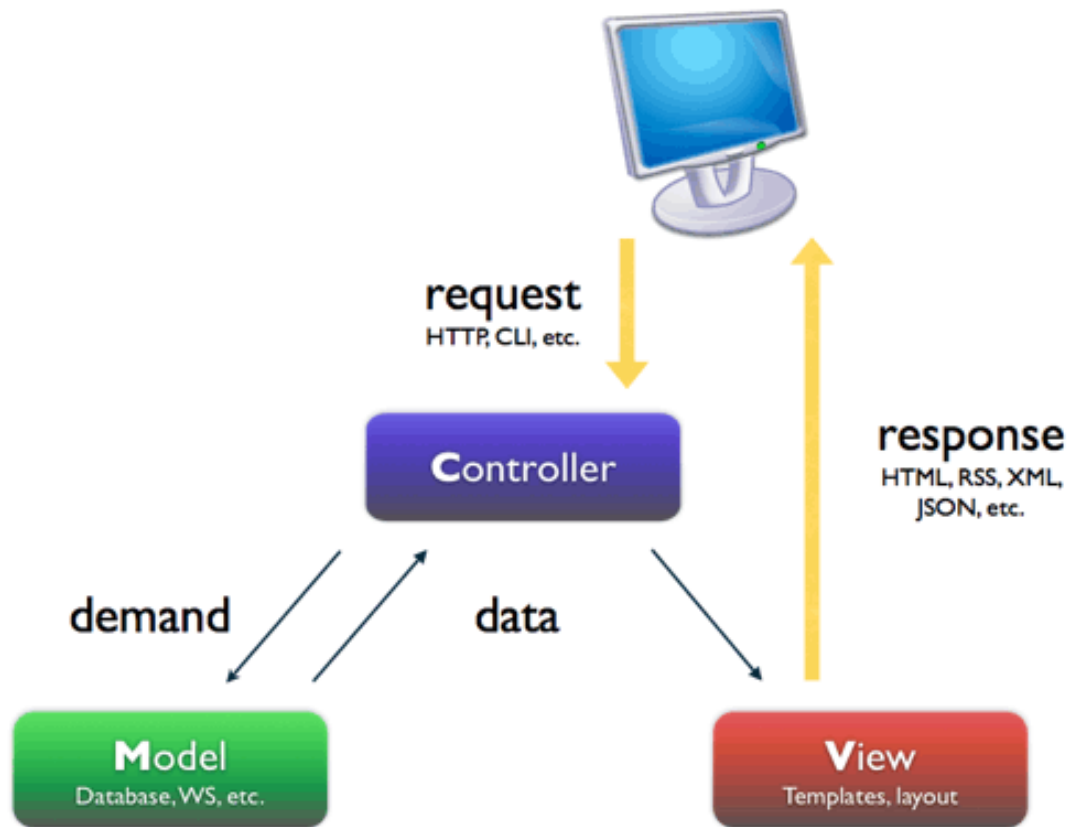
Channel Message Queues

Channel	Depth	Memory + Disk	In-Flight	Deferred	Requeued	Timed Out	Messages	Connections
Actions	0	0 + 0	0	0	1	1	93,831,276	1
Events	0	0 + 0	10	0	9	9	93,831,276	1
LTV	0	0 + 0	0	0	1	1	61,394,872	1
Profiles	0	0 + 0	0	0	0	0	93,831,276	1

*nsq message processing system (nsq.io (<http://nsq.io>))

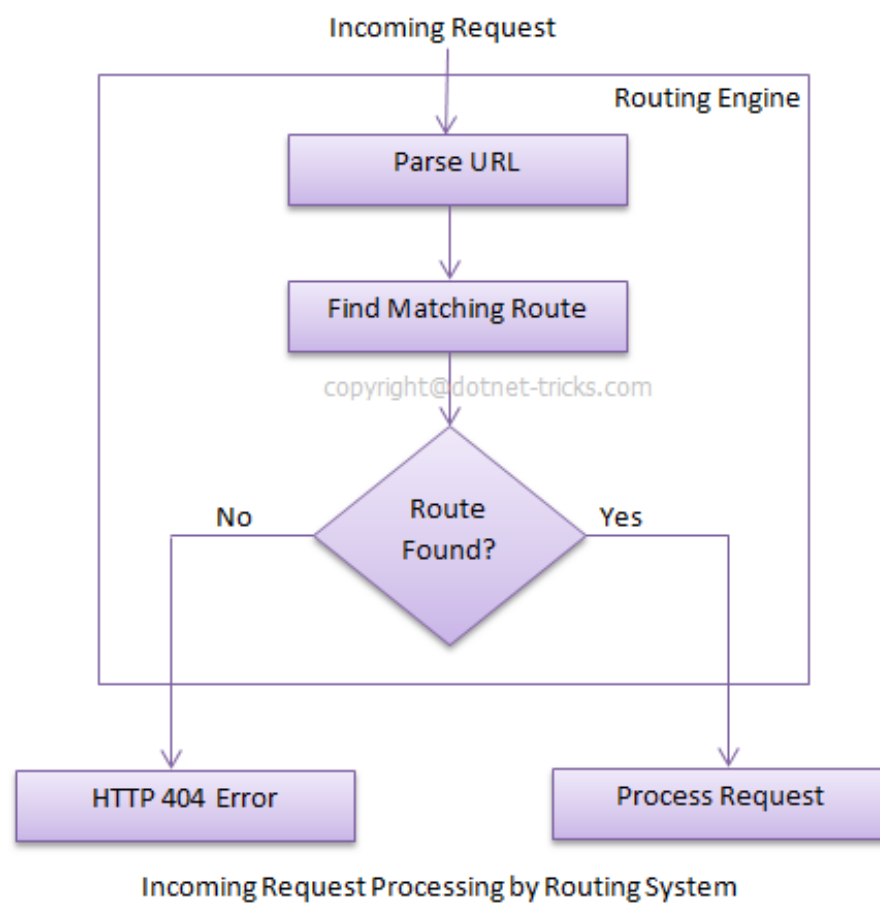
Основной подход к разработке - архитектурный паттерн MVC

Model-View-Controller



- Для чего? Для разделения графического интерфейса от бизнес-логики

Роутинг



- Классический пример: <http://example.com/controller/action>

Роутинг - примеры реализации в разных библиотеках

- **net/http:**

```
func statusHandler(db *sql.DB) http.HandlerFunc {  
    return func(wr http.ResponseWriter, req *http.Request) {  
        ...  
    }  
}  
  
http.Handle("/status", statusHandler(db))  
http.Handle("/import_news", importNewsHandler(db))  
...  
http.ListenAndServe(":8080", nil)
```

- **Gin Framework**, один из самых быстрых роутинов:

github.com/gin-gonic/gin/blob/master/README.md (<https://github.com/gin-gonic/gin/blob/master/README.md>)

```
router := gin.Default()  
router.GET("/someGet", getting)  
router.POST("/somePost", posting)  
...  
router.Run(":8080")
```

Контроллер (Controller)

- Набор функций для генерации ответа пользователю
- Совмещает в себе вызов функций бизнес-логики и формирование представления
- Одна функция == action (<http://example.com/controller/action>)
- Пример (без использования сторонних фреймворков):

```
type Controller struct {  
    ...  
}  
  
func (c *Controller) Action(wr http.ResponseWriter, req *http.Request) {  
    resp := interface{}  
  
    bytes, err := json.Marshal(resp)  
    if err != nil {  
        writeErr(wr, err)  
        return  
    }  
    wr.Write(bytes)  
}
```

Контроллер для Gin Framework

- Не сильно отличается
- Да и в других фреймворках практически тоже самое

```
type Controller struct {  
    ...  
}  
  
func (c *Controller) Action(ctx *gin.Context) {  
    ...  
}
```

- Есть готовый мощный объект **gin.Context**, который передается в методы-обработчики

Модель (Model)

- Модель - это бизнес-логика приложения
- Модель обладает знаниями о себе самой и не знает о контроллерах и представлениях
- Для некоторых проектов модель — это просто слой данных (база данных, XML-файл, ...)
- Для других проектов модель — это менеджер базы данных, набор объектов или просто логика приложения

```
type User struct {  
    Id    int    `bson:"_id"`  
    Login string `bson:"login"`  
}  
  
// Gets user from database by login  
func GetUser(db *mongo.Database, login string) (User, error) {  
    var result User  
    err := db.C(usersCollName).Find(bson.M{"login": login}).One(&result)  
    return result, err  
}
```

Представление (View)

- В представлении реализуется визуализация данных, которые получаются от модели любым способом
- HTML, JSON, XML, RSS ...

```
// Gin framework
router.GET("/index", func(c *gin.Context) {
    c.HTML(http.StatusOK, "index.tmpl", gin.H{
        "title": "Main website",
    })
})

<!-- Index template (View) -->
<html>
    <h1>
        {{ .Title }}
    </h1>
</html>
```

Шаблоны (Templates)

- Есть две мощные библиотеки:
- golang.org/pkg/text/template (<https://golang.org/pkg/text/template>)
- golang.org/pkg/html/template (<https://golang.org/pkg/html/template>)
- Есть множество вариантов подстановок (условия, циклы, вызовы функций внутри шаблона)

```
<!-- Например, цикл для формирования списка данных -->
{{ range .Pages }}
    <li>{{ $.Name }}</li>
{{ end }}
```

Генерация готового представления (view) из шаблона

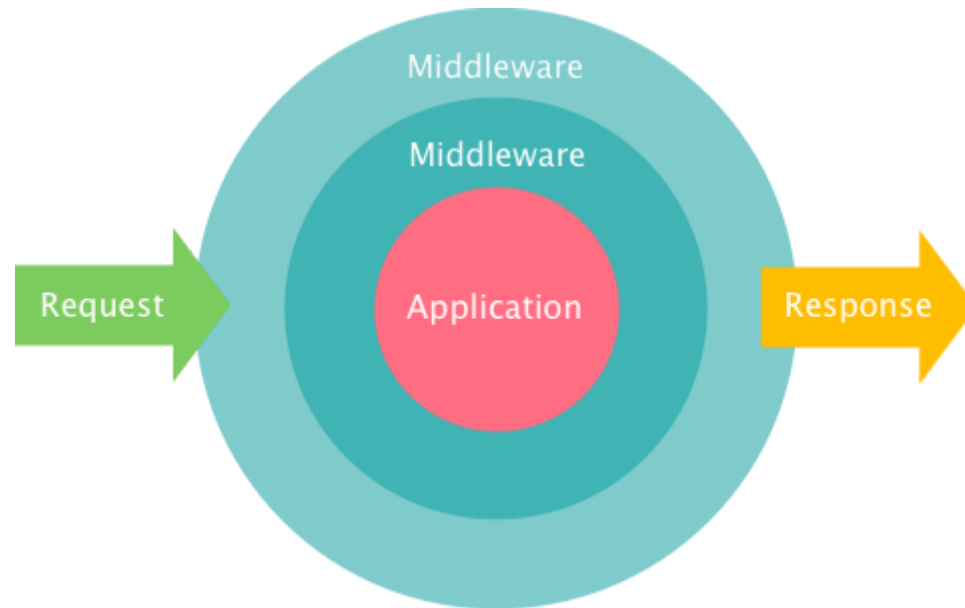
```
import (  
    "html/template"  
    "os"  
)  
  
// Здесь просто переменная, можно грузить из файла  
var tmpl = `  
<html><h1>{{ .Title }}</h1></html>`  
  
data := struct{  
    Title string  
}{ "Main page"  
  
t := template.New("name")  
t, err := t.Parse(tmpl)  
if err != nil {  
    panic(err)  
}  
  
err = t.Execute(os.Stdout, data)  
if err != nil {  
    panic(err)  
}
```


Есть другой вариант - использовать AJAX для отображения данных на странице

- Asynchronous Javascript and XML
- Появляется два хендлера: для статического HTML и для данных в формате JSON
- Вызываем скриптом специальный хендлер, который возвращает JSON
- Изменяем элементы в DOM (Document Object Model)

```
function updateSubscribers() {  
    var url = "subscribers?startDate=" + startDate + "&endDate=" + endDate +  
        "&curDate=" + curDate;  
  
    $.get(url, function(response) {  
        var attr = "data-val";  
        $("#total").attr(attr, response.total);  
        $("#active").attr(attr, response.active);  
    });  
}
```

Middleware



- Цепочка дополнительных обработчиков в конвейере обработки запроса
- Активно применяется в **Gin Framework**, **Martini Framework**, **Revel Framework**

Примеры middleware

- Gin Framework

```
router := gin.Default()
router.GET("/benchmark", MyBenchLogger(), benchEndpoint)
```

- Martini Framework

```
email = controllers.EmailController{}

func (ec *EmailController) Index(rndr render.Render, req *http.Request, sess *session.Session) {
    data, err := dbmodels.GetSites(dbmodels.MongoDB, sess.User)
    if err != nil {
        Error(rndr, req, "emailIndex", "Unable to get sites", err)
        return
    }
    page := models.NewWebPage(req, "Emails", "emailIndex", data)
    rndr.HTML(200, "email_index", page, GetHtmlOptions(sess))
}

m := martini.Classic()
m.Group("/email", func(router martini.Router) {
    router.Get("/clone", loginProtect, mngrProtect, siteProtect, email.Clone)
    router.Get("/index", loginProtect, mngrProtect, email.Index)
})
```

Задание

- Создать простой веб-сайт (название **website**) на основе библиотеки **net/http**
- Без использования фреймворков, только **net/http**
- На сайте одна страница которая отображает список пользователей в виде таблицы
- Заголовки таблицы: "First Name", "Email", "Username"
- Необходимо использовать **MVC подход**
- Данные в модели заполняются программно, т.е. где-то хардкодом забиваются
- Выводить таблицу с использованием шаблона (Templates)

^ Id	Name	Login	Role
3	Vladimir Ananyev	vladimir.ananyev	Administrator
5	Artem Kukhareenko	artem.kukhareenko	Administrator
6	Tatyana Murzova	tatyana.murzova	Administrator

Thank you

11 ноября 2016

Владимир Ананьев

АО "Программный Регион"

vladimir.ananyev@regium.com (mailto:vladimir.ananyev@regium.com)

