

使用 GitHub Actions 对 GitHub Profile 个人主页进行自动更新

作者: [88250](#)

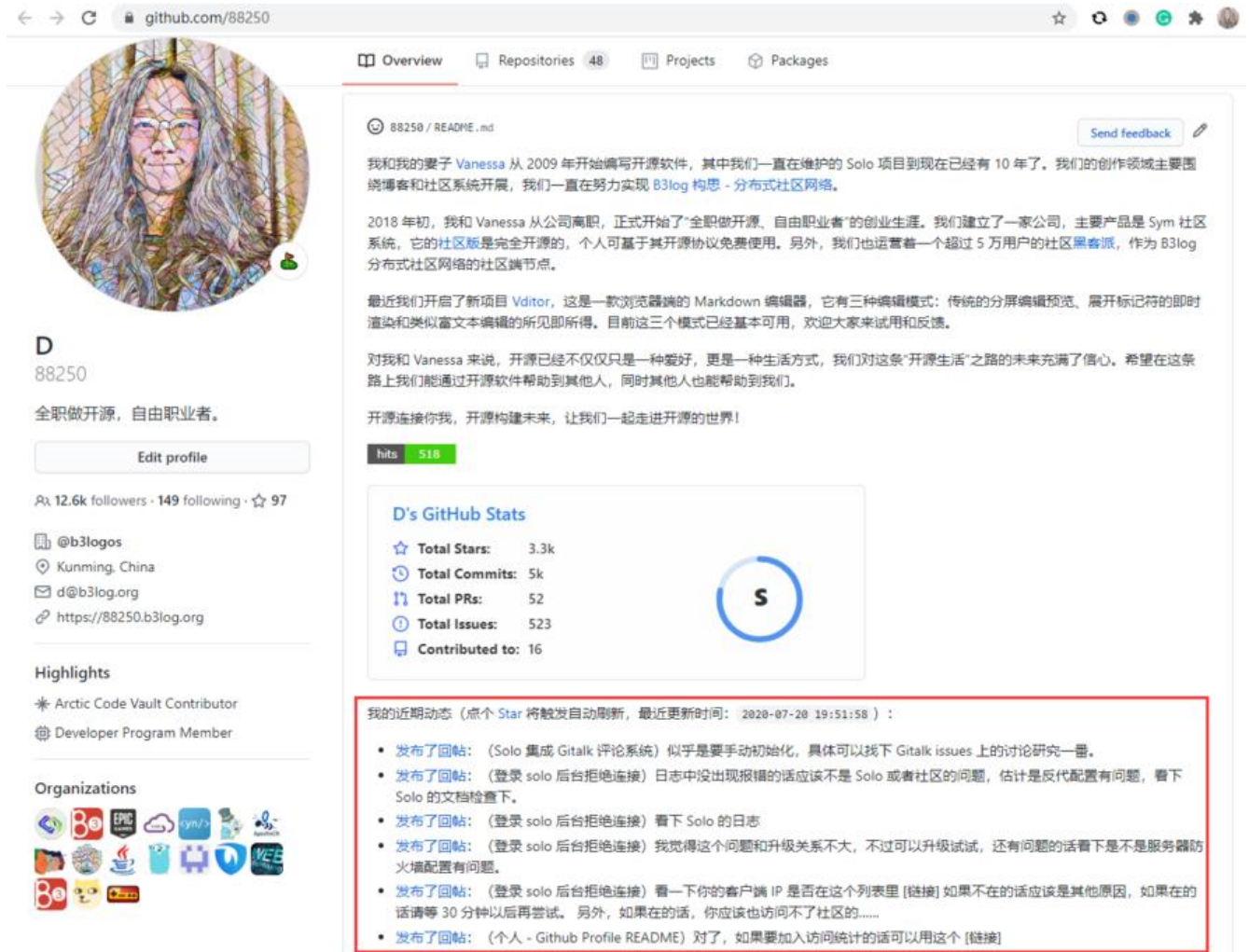
原文链接: <https://ld246.com/article/1595248018192>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

实际效果

先看效果 <https://github.com/88250>



88250 / README .md

我和我的妻子 Vanessa 从 2009 年开始编写开源软件，其中我们一直在维护的 Solo 项目到现在已经有 10 年了。我们的创作领域主要围绕博客和社区系统开展，我们一直在努力实现 83log 构思 - 分布式社区网络。

2018 年初，我和 Vanessa 从公司离职，正式开始了“全职做开源、自由职业者”的创业生涯。我们建立了一家公司，主要产品是 Sym 社区系统，它的社区版是完全开源的，个人可基于其开源协议免费使用。另外，我们也运营着一个超过 5 万用户的社区黑客派，作为 83log 分布式社区网络的社区端节点。

最近我们开启了新项目 Vditor，这是一款浏览器端的 Markdown 编辑器，它有三种编辑模式：传统的分屏编辑预览、展开标记符的即时渲染和类似富文本编辑的所见即所得。目前这三个模式已经基本可用，欢迎大家来试用和反馈。

对我和 Vanessa 来说，开源已经不仅仅只是一种爱好，更是一种生活方式，我们对这条“开源生活”之路的未来充满了信心。希望在这条路上我们能通过开源软件帮助到其他人，同时其他人也能帮助到我们。

开源连接你我，开源构建未来，让我们一起走进开源的世界！

hits 538

D's GitHub Stats

- Total Stars: 3.3k
- Total Commits: 5k
- Total PRs: 52
- Total Issues: 523
- Contributed to: 16

我的近期动态 (点个 Star 将触发自动刷新, 最近更新时间: 2020-07-20 19:51:58) :

- 发布了回帖: (Solo 集成 Gitalk 评论系统) 似乎是要手动初始化, 具体可以找下 Gitalk issues 上的讨论研究一番。
- 发布了回帖: (登录 solo 后台拒绝连接) 日志中没出现报错的话应该不是 Solo 或者社区的问题, 估计是反代配置有问题, 看下 Solo 的文档检查下。
- 发布了回帖: (登录 solo 后台拒绝连接) 看下 Solo 的日志
- 发布了回帖: (登录 solo 后台拒绝连接) 我觉得这个问题和升级关系不大, 不过可以升级试试, 还有问题的话看下是不是服务器防火墙配置有问题。
- 发布了回帖: (登录 solo 后台拒绝连接) 看一下你的客户端 IP 是否在这个列表里 [链接] 如果不在的话应该是其他原因, 如果在的话请等 30 分钟以后再尝试。另外, 如果在的话, 你也应该访问不了社区的.....
- 发布了回帖: (个人 - Github Profile README) 对了, 如果要加入访问统计的话可以用这个 [链接]

红框部分是通过 GitHub Actions 自动更新的，内容来源由社区 API 获取用户近期动态列表提供。当人 Star 我的个人主页仓库 <https://github.com/88250/88250> 时就会触发该 GitHub Action 从而完 README.md 的自动更新，最终在个人主页页面就可以看到以上效果了。

代码实现

最新代码请以我仓库实际代码为准，以下代码仅为了说明原理。

Action 配置如下：

name: Update events

on:

watch:

types: started

jobs:

build:

runs-on: ubuntu-latest

```

steps:
- name: Check out repo
  uses: actions/checkout@v2
- uses: actions/setup-go@v2
- name: Fetch events
  run: go run main.go
- name: Commit and push
  run: |-
    git config --global user.email "bot@github.com" && git config --global user.name "Bot"
    git diff
    git add . && git commit -m ":memo: 更新自述" || exit 0
    git push

```

除了 Star 触发，也可以有其他触发方式，比如代码提交、外部接口或者定时等，具体可参考 [Events that trigger workflows](#)。

主要逻辑是通过 golang 实现，获取数据并更新 README.md 文件，该程序代码如下：

```

package main

import (
    "bytes"
    "crypto/tls"
    "fmt"
    "io/ioutil"
    "net/http"
    "os"
    "time"

    "github.com/88250/gulu"
    "github.com/parnurzeal/gorequest"
)

var logger = gulu.Log.NewLogger(os.Stdout)

const (
    githubUserName = "88250"
    hacpaiUserName = "88250"
)

func main() {
    result := map[string]interface{}{}
    response, data, errors := gorequest.New().TLSClientConfig(&tls.Config{InsecureSkipVerify: true}).
        Get("https://hacpai.com/api/v2/user/" + hacpaiUserName + "/events?size=8").Timeout(7 * time.Second).
        Set("User-Agent", "Profile Bot; +https://github.com/" + githubUserName + "/" + githubUserName).EndStruct(&result)
    if nil != errors || http.StatusOK != response.StatusCode {
        logger.Fatalf("fetch events failed: %+v, %s", errors, data)
    }
    if 0 != result["code"].(float64) {
        logger.Fatalf("fetch events failed: %s", data)
    }
}

```

```

}

buf := &bytes.Buffer{}
buf.WriteString("\n\n")
updated := time.Now().Format("2006-01-02 15:04:05")
buf.WriteString("我的近期动态 (点个 [Star](https://github.com/" + githubUserName + "/" +
githubUserName + ") 将触发自动刷新, 最近更新时间: ` ` + updated + "` ) : \n\n")
for _, event := range result["data"].([]interface{}) {
    evt := event.(map[string]interface{})
    operation := evt["operation"].(string)
    title := evt["title"].(string)
    url := evt["url"].(string)
    content := evt["content"].(string)
    buf.WriteString("* [" + operation + "]" + url + ": (" + title + ") " + content + "\n")
}
buf.WriteString("\n")

fmt.Println(buf.String())

readme, err := ioutil.ReadFile("README.md")
if nil != err {
    logger.Fatalf("read README.md failed: %s", data)
}

startFlag := []byte("<!--events start -->")
beforeStart := readme[:bytes.Index(readme, startFlag)+len(startFlag)]
newBeforeStart := make([]byte, len(beforeStart))
copy(newBeforeStart, beforeStart)
endFlag := []byte("<!--events end -->")
afterEnd := readme[bytes.Index(readme, endFlag):]
newAfterEnd := make([]byte, len(afterEnd))
copy(newAfterEnd, afterEnd)
newReadme := append(newBeforeStart, buf.Bytes()...)
newReadme = append(newReadme, newAfterEnd...)
if err := ioutil.WriteFile("README.md", newReadme, 0644); nil != err {
    logger.Fatalf("write README.md failed: %s", data)
}
}
}

```

大家也可以用上

如果你也想将社区动态放到 GitHub 个人主页上, 可以 Fork 我的仓库, 然后:

1. 修改 README.md, 将内容改为自己的, `<!--events start -->` 和 `<!--events end -->` 中间留, 大概就是这样:

```

...
你的 README 主体内容
...
<!--events start -->
<!--events end -->

```

2. 修改 main.go 中 `githubUserName` 和 `hacpaiUserName` 常量, 都改为对应的用户名
3. 提交代码推送仓库

然后自己点下 Star 测试看看，可以在自己仓库的 Actions 上看到执行过程输出 ([我的实例](#))，如果报错的话请跟帖反馈。

Just for fun trollface