一个 golang HTTP 客户端

作者：lhlyu
原文链接：https://ld246.com/article/1587095693333
来源网站：链滴

基于gorequest进行改造，支持异步请求，链式处理响应，文件保存...

项目地址

链接

例子

```
package test

import (
    "github.com/lhlyu/request/v2"
    "testing"
    "time"
)

const (
    apiUrl    = "https://api.github.com/users/lhlyu"
    picUrl    = "https://cdn.jsdelivr.net/gh/lhlyu/pb@master/b/43.jpg"
)


func TestNew(t *testing.T) {
    body := request.New().
        Debug().
        Get(apiUrl).GetBodyString()
    t.Log(body)
}

type User struct {
    Id int `json:"id"`
}

func TestSend(t *testing.T) {
    body := request.New().
        Timeout(time.Second * 10).
        Send(`{"id":123}`).
        Send(&User{1}).
        Send("id=1").
        Post("http://localhost:8080/article/%d",1).GetBodyString()
    t.Log(body)
}

func dosomething(){
    time.Sleep(time.Second)
}

// 异步请求
func TestAsynch(t *testing.T) {
    r := request.New().
        Asynch().   // 开启异步
        Get(apiUrl)
```

```
    // 可以做别的事
    dosomething()

    // 等待
    body := r.Await().GetBodyString()

    t.Log(body)
}

// 保存文件
func TestSave(t *testing.T) {
    err := request.New().
        Get(picUrl).Save("./test.jpg")
    t.Log(err)
}

// 链式调用
func TestThen(t *testing.T) {
    request.New().
        Get(apiUrl).Then(func(r request.Receiver) {
        t.Log(r.GetStatus())
    }).Then(func(r request.Receiver) {
        t.Log(r.GetBodyString())
    }).Then(func(r request.Receiver) {
        u := &User{}
        r.BodyUnmarshal(u)
        t.Log(u.Id)
        if u.Id > 100000{
            r.Stop()  // 会导致后面的操作停止
        }
    }).Then(func(r request.Receiver) {
        // 正则匹配
        vals := r.BodyCompile("\\d+")
        for _, v := range vals {
            t.Log(v)
        }
    })
}
```

使用

- 建议golang版本1.13+
- 设置代理

```
go env -w GOPROXY=https://goproxy.cn,direct
```

- 下载

```
go get -v github.com/lhlyu/request/v2
```

请求接口定义

```
// 请求体接口定义
type Agent interface {
    // 开启debug
    Debug() Agent
    // 克隆
    Clone() Agent
    // 设置超时
    Timeout(timeout time.Duration) Agent
    // 是否打印curl命令
    Curl() Agent
    // 开启异步
    Asynch() Agent
    // 每次请求前都清空Agent
    SetDoNotClearSuperAgent(enable bool) Agent
    // 设置请求头
    SetHeader(param string, value string) Agent
    // 添加请求头
    AddHeader(param string, value string) Agent
    // 设置重试： 重试次数,间隔时间,期望返回的状态码列表
    Retry(retryerCount int, retryerTime time.Duration, statusCode ...int) Agent
    // 设置auth
    SetBasicAuth(username string, password string) Agent
    // 添加cookie
    AddCookie(c *http.Cookie) Agent
    // 添加简单cookie
    AddSimpleCookie(name, val string) Agent
    // 添加cookies
    AddCookies(cookies []*http.Cookie) Agent
    // 设置请求数据类型
    Type(typeStr string) Agent
    // 设置url参数
    Query(content interface{}) Agent
    // 添加url参数
    AddParam(key string, value string) Agent
    // 传输层安全协议配置 http2
    TLSClientConfig(config *tls.Config) Agent
    // 设置代理地址
    Proxy(proxyUrl string) Agent
    // 设置重定向策略
    RedirectPolicy(policy func(req *http.Request, via []*http.Request) error) Agent
    // 发送内容，一般放在body,自动判断类型
    Send(content interface{}) Agent
    // 指定发送切片
    SendSlice(content []interface{}) Agent
    // 指定发送map
    SendMap(content interface{}) Agent
    // 指定发送struct
    SendStruct(content interface{}) Agent
    // 指定发送string
    SendString(content string) Agent
    // 发送文件,允许输入文件路径，文件流, [文件名,Field]
    SendFile(file interface{}, args ...string) Agent

    // 通用请求方法
```

```
    Http(method, targetUrl string, a ...interface{}) Receiver
    // GET请求
    Get(targetUrl string, a ...interface{}) Receiver
    // POST请求
    Post(targetUrl string, a ...interface{}) Receiver
    // HEAD请求
    Head(targetUrl string, a ...interface{}) Receiver
    // PUT请求
    Put(targetUrl string, a ...interface{}) Receiver
    // DELETE请求
    Delete(targetUrl string, a ...interface{}) Receiver
    // PATCH请求
    Patch(targetUrl string, a ...interface{}) Receiver
    // OPTIONS请求
    Options(targetUrl string, a ...interface{}) Receiver

    // 返回curl
    AsCurlCommand() (string, error)
}
```

- 响应接口定义

```
// 响应接口定义
type Receiver interface {
    // 获取原生响应体
    GetResponse() *http.Response
    // 获取body内容
    GetBody() []byte
    // 获取body内容
    GetBodyString() string
    // body内容转对象
    BodyUnmarshal(v interface{}) error
    // 正则匹配body内容
    BodyCompile(pattern string) []string
    // 获取响应状态码，没有则返回 -1
    GetStatus() int
    // 异步等待响应结果返回
    Await() Receiver
    // 保存到文件
    Save(fl interface{}) error
    // 后续操作
    Then(fn func(r Receiver)) Receiver
    // 停止后续操作
    Stop()
    // 获取最后一个错误
    Error() error
    // 获取所有错误
    Errors() []error
}
```