gtoken

https://github.com/goflyfox/gtoken

Content Menu

```
        //
        s.Group("/", func(group *ghttp.RouterGroup) {
                group.Middleware(CORS)

                //
                group.ALL("/hello", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("hello"))
                })
        })

        //
        loginFunc := Login
        // gtoken
        gfToken := &gtoken.GfToken{
                ServerName:       TestServerName,
                LoginPath:        "/login",
                LoginBeforeFunc:  loginFunc,
                LogoutPath:       "/user/logout",
                AuthExcludePaths: g.SliceStr{"/user/info", "/system/user
/info"}, //  /user/info,/system/user/info,/system/user,
                MultiLogin:       g.Config().GetBool("gToken.MultiLogin"),
        }
        s.Group("/", func(group *ghttp.RouterGroup) {
                group.Middleware(CORS)
                gfToken.Middleware(group)

                group.ALL("/system/user", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("system user"))
                })
                group.ALL("/user/data", func(r *ghttp.Request) {
                        r.Response.WriteJson(gfToken.GetTokenData(r))
                })
                group.ALL("/user/info", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("user info"))
                })
                group.ALL("/system/user/info", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("system user
info"))
                })
        })

        // gtoken
        gfAdminToken := &gtoken.GfToken{
                ServerName: TestServerName,
                //Timeout:         10 * 1000,
                LoginPath:        "/login",
                LoginBeforeFunc:  loginFunc,
                LogoutPath:       "/user/logout",
                AuthExcludePaths: g.SliceStr{"/admin/user/info", "/admin
/system/user/info"}, //  /user/info,/system/user/info,/system/user,
                MultiLogin:       g.Config().GetBool("gToken.MultiLogin"),
        }
        s.Group("/admin", func(group *ghttp.RouterGroup) {
                group.Middleware(CORS)
                gfAdminToken.Middleware(group)

                group.ALL("/system/user", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("system user"))
                })
                group.ALL("/user/info", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("user info"))
                })
                group.ALL("/system/user/info", func(r *ghttp.Request) {
                        r.Response.WriteJson(gtoken.Succ("system user
info"))
                })
        })
```

gmanager

https://github.com/goflyfox/gmanager

```
        urlPath := g.Config().GetString("url-path")
        s := g.Server()

        s.Group(urlPath+"/", func(group *ghttp.RouterGroup) {
                //
                group.Middleware(func(r *ghttp.Request) {
                        r.Response.CORSDefault()
                        r.Middleware.Next()
                })
                //
                group.Middleware(middle.MiddlewareLog)
                //
                group.Middleware(middle.MiddlewareCommon)

                //
                group.ALL("/", common.Login)
                group.ALL("/main.html", common.Index)
                group.ALL("/login", common.Login)

                group.ALL("/welcome", common.Welcome)
                group.ALL("/admin/welcome.html", common.Welcome)

                // gtoken
                base.Token = &gtoken.GfToken{
                        //Timeout:         10 * 1000,
                        CacheMode:        g.Config().GetInt8("gtoken.cache-
mode"),
                        MultiLogin:       g.Config().GetBool("gtoken.multi-
login"),
                        LoginPath:        "/login/submit",
                        LoginBeforeFunc:  common.LoginSubmit,
                        LogoutPath:       "/user/logout",
                        LogoutBeforeFunc: common.LogoutBefore,
                        AuthPaths:        g.SliceStr{"/user", "/system"},
                        GlobalMiddleware: true,
                        AuthBeforeFunc: func(r *ghttp.Request) bool {
                                //
                                if r.IsFileRequest() {
                                        return false
                                }

                                if strings.HasSuffix(r.URL.Path, "index") {
                                        return false
                                }

                                return true
                        },
                }
                //
                group.Group(urlPath+"/system", func(group *ghttp.
RouterGroup) {
                        // gtoken
                        base.Token.Middleware(group)

                        //
                        userAction := new(user.Action)
                        group.ALL("user", userAction)
                        group.GET("/user/get/{id}", userAction.Get)
                        group.ALL("user/delete/{id}", userAction.Delete)

                        departAction := new(department.Action)
                        group.ALL("department", departAction)
                        group.GET("/department/get/{id}", departAction.Get)
                        group.ALL("/department/delete/{id}", departAction.
```

```
Delete)

                        logAction := new(log.Action)
                        group.ALL("log", logAction)
                        group.GET("/log/get/{id}", logAction.Get)
                        group.ALL("/log/delete/{id}", logAction.Delete)

                        menuAction := new(menu.Action)
                        group.ALL("menu", menuAction)
                        group.GET("/menu/get/{id}", menuAction.Get)
                        group.ALL("/menu/delete/{id}", menuAction.Delete)

                        roleAction := new(role.Action)
                        group.ALL("role", roleAction)
                        group.GET("/role/get/{id}", roleAction.Get)
                        group.ALL("/role/delete/{id}", roleAction.Delete)

                        configAction := new(config.Action)
                        group.ALL("config", configAction)
                        group.GET("/config/get/{id}", configAction.Get)
                        group.ALL("/config/delete/{id}", configAction.
Delete)
                })
        })
```

gfast

https://github.com/tiger1103/gfast

```
func initAdminGfToken() {
        //
        service.AdminMultiLogin = g.Cfg().GetBool("gToken.MultiLogin")
        AdminGfToken = &gtoken.GfToken{
                CacheMode:        g.Cfg().GetInt8("gToken.CacheMode"),
                CacheKey:         g.Cfg().GetString("gToken.CacheKey"),
                Timeout:          g.Cfg().GetInt("gToken.Timeout"),
                MaxRefresh:       g.Cfg().GetInt("gToken.MaxRefresh"),
                TokenDelimiter:   g.Cfg().GetString("gToken.
TokenDelimiter"),
                EncryptKey:       g.Cfg().GetBytes("gToken.EncryptKey"),
                AuthFailMsg:      g.Cfg().GetString("gToken.AuthFailMsg"),
                MultiLogin:       service.AdminMultiLogin,
                LoginPath:        "/sysLogin/login",
                LoginBeforeFunc:  service.AdminLogin,
                LoginAfterFunc:   service.LoginAfter,
                LogoutPath:       "/sysLogin/logout",
                AuthPaths:        g.SliceStr{"/system/*"},
                AuthAfterFunc:    service.AuthAfterFunc,
                LogoutBeforeFunc: service.LoginOut,
        }
        AdminGfToken.Start()
}
```