课程安排，请关注**微信公众平台**或者**官方微博**

编程语言： Golang 与 html5

编程工具： Goland 和 HBuilder

预计平均一周左右更新一或二节课程

授人以鱼，不如授人以渔业。

大家好，
欢迎来到 **字节教育** 课程的学习。
字节教育官网：**www.ByteEdu.Com**
腾讯课堂地址：**Gopher.ke.qq.Com**
技术交流群 ： 221 273 219
微信公众号 ：**Golang 语言社区**
微信服务号 ：**Golang 技术社区**

目录：

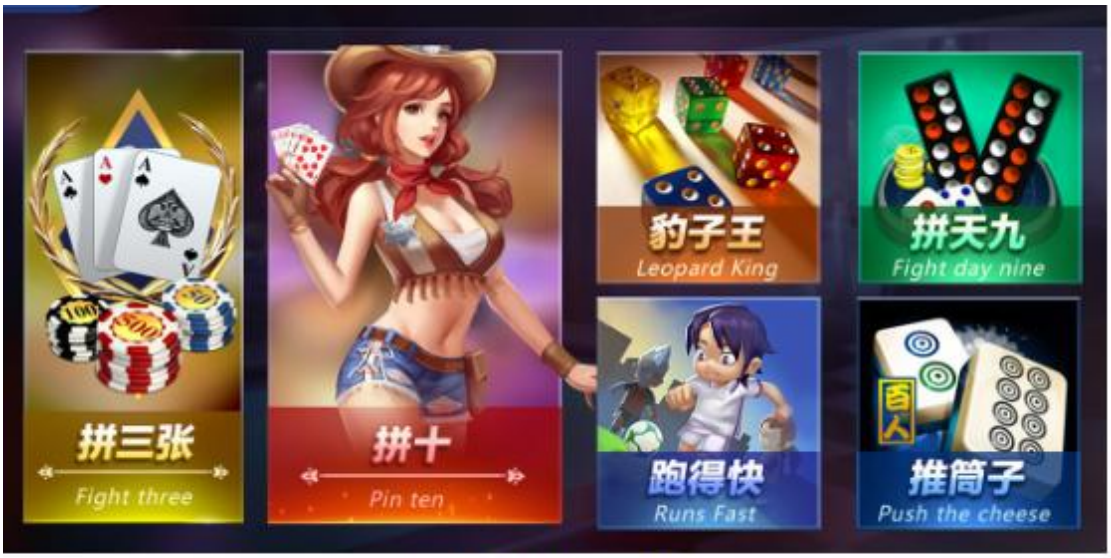
第一季 LollipopGo 开源架构实战项目

第一节　框架 gate 及 network 学习

1、LollipopGo 框架可以做什么类型游戏？

<1> 三消类型游戏

<2> 棋牌游戏



手机棋牌开发

<3> 挂机游戏

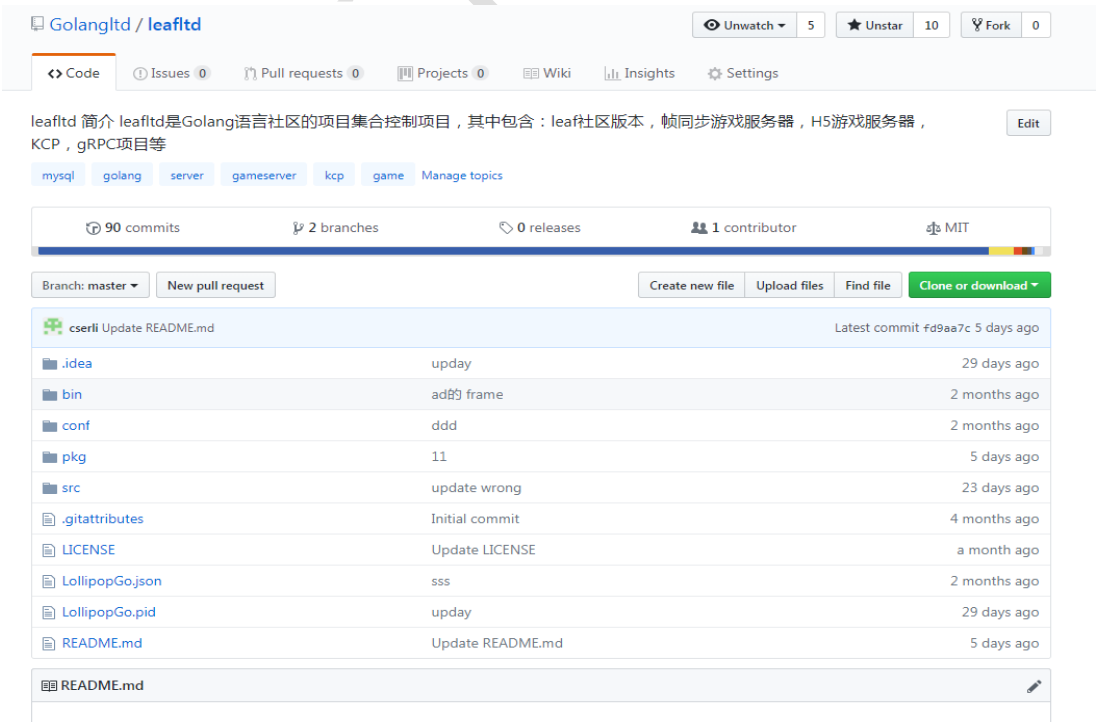
<4>SLG 策略游戏

《王国纪元》

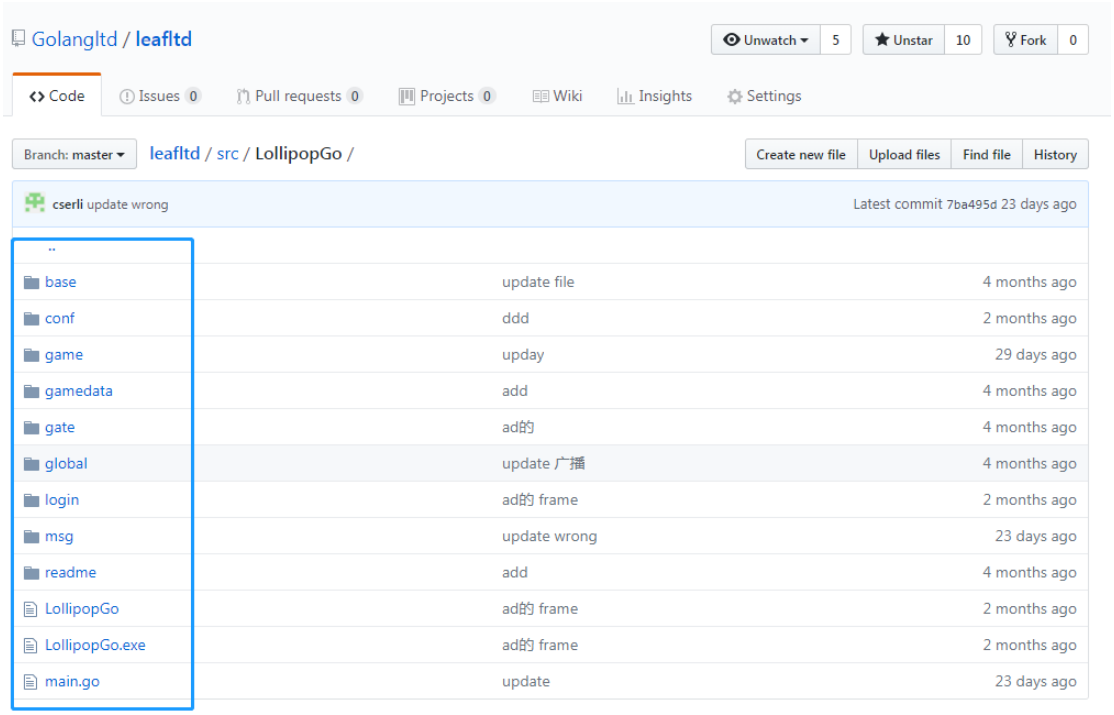
总结：LollipopGo 社区框架，设计思想来自于 Leaf+OnlineGo 设计思想，轻度游戏架构设计；实时低延迟游戏服务器数据传输，设计：每秒 20 帧；同时此框架针对同步服务器初步设计而成，本系列教程主要是针对社区的 LollipopGo 1.02 版本 leaf 的超级版本而做的课程，设计流程和 leaf 吻合；只是做了初步的部分数据拓展及优化。

2、框架项目的 github 地址及工程目录结构说明

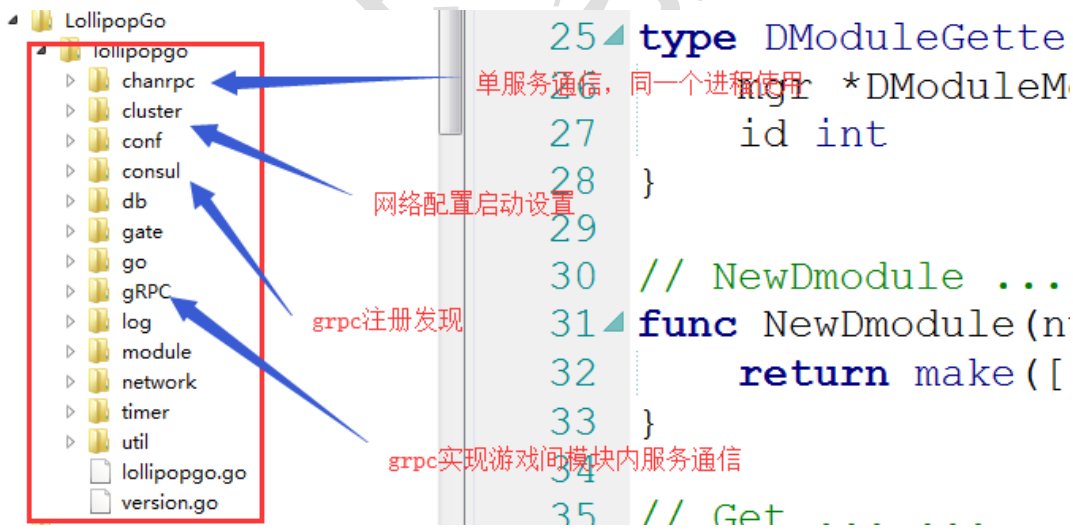
地址：
https://github.com/Golangltd/leafltd



游戏服务项目地址：



框架结构说明：



3、字节教育 开设本系列实战课程的目的

首先感谢大家的支持，目前字节属于成立初期，所有的课程都再慢慢制定中，由于我们课程老师的经验不足，可能会导致课程的录制不够好，我们针对学员的反馈意见已经做了部分的调整；翻录不够好的课程，一定要达到一个可以让学员接受的课程质量。

字节教育，成立本系列课程主要是想帮助有基础想进入游戏开发领域的学员一起开发

他们的游戏开发之旅。不过我们课程主要是偏向后端，所有的前端都是带过的，这一点希望报名的学员理解。

如何学习好 LollipopGo 项目架构？

<1>. 认真听课，课后复习

<2>. 课下自己实践

<3>. 社区交流群交流

<4>. 自己做个小项目使用学到的技术

4、提供给大家学习交流的平台

<1> 社区 H5 应用 问答系统：

进入方式：Golang 技术社区 菜单栏

<2> 社区论坛 www.Golang.Ltd

5、gate 讲解

agent.go 文件：

```
type Agent interface {
        WriteMsg(msg interface{})
        LocalAddr() net.Addr
        RemoteAddr() net.Addr
        Close()
        Destroy()
        UserData() interface{}
        SetUserData(data interface{})
}

// 在线玩家的数据的结构体
type OnlineUser struct {
        Connection Agent                    // 链接的信息
        StrMD5     string                   // 用的 UID 标示
        MapSafe    *concurrent.ConcurrentMap // 并发安全的 map
}
```

Gate.go 文件：

```
type Gate struct {
        MaxConnNum      int
        PendingWriteNum int
        MaxMsgLen       uint32
        Processor       network.Processor
        AgentChanRPC    *chanrpc.Server

        // websocket
        WSAddr      string
        HTTPTimeout time.Duration
        CertFile    string
        KeyFile     string

        // tcp
        TCPAddr      string
        LenMsgLen    int
        LittleEndian bool
}
```

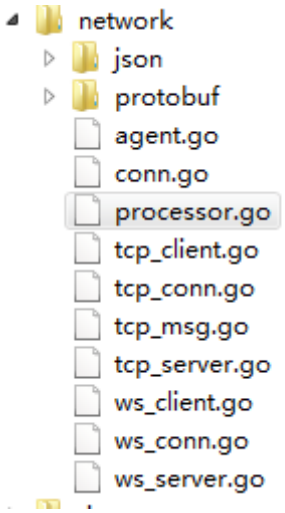
```
func (gate *Gate) Run(closeSig chan bool) {
	var wsServer *network.WSServer
	if gate.WSAddr != "" {
		wsServer = new(network.WSServer)
		wsServer.Addr = gate.WSAddr
		wsServer.MaxConnNum = gate.MaxConnNum
		wsServer.PendingWriteNum = gate.PendingWriteNum
		wsServer.MaxMsgLen = gate.MaxMsgLen
		wsServer.HTTPTimeout = gate.HTTPTimeout
		wsServer.CertFile = gate.CertFile
		wsServer.KeyFile = gate.KeyFile
		wsServer.NewAgent = func(conn *network.WSConn) network.Agent {
			a := &agent{conn: conn, gate: gate}
			if gate.AgentChanRPC != nil {
				gate.AgentChanRPC.Go("NewAgent", a)
			}
			return a
		}
	}

	var tcpServer *network.TCPServer
	if gate.TCPAddr != "" {
		tcpServer = new(network.TCPServer)
		tcpServer.Addr = gate.TCPAddr
		tcpServer.MaxConnNum = gate.MaxConnNum
		tcpServer.PendingWriteNum = gate.PendingWriteNum
		tcpServer.LenMsgLen = gate.LenMsgLen
		tcpServer.MaxMsgLen = gate.MaxMsgLen
		tcpServer.LittleEndian = gate.LittleEndian
		tcpServer.NewAgent = func(conn *network.TCPConn) network.Agent {
			a := &agent{conn: conn, gate: gate}
			if gate.AgentChanRPC != nil {
				gate.AgentChanRPC.Go("NewAgent", a)
			}
			return a
		}
	}

	if wsServer != nil {
		wsServer.Start()
	}
	if tcpServer != nil {
		tcpServer.Start()
```

```
        }
        <-closeSig
        if wsServer != nil {
                wsServer.Close()
        }
        if tcpServer != nil {
                tcpServer.Close()
        }
}

func (gate *Gate) OnDestroy() {}

type agent struct {
        conn     network.Conn
        gate     *Gate
        userData interface{}
}

func (a *agent) Run() {
        for {
                data, err := a.conn.ReadMsg()
                if err != nil {
                        log.Debug("read message: %v", err)
                        break
                }

                if a.gate.Processor != nil {
                        msg, err := a.gate.Processor.Unmarshal(data)
                        if err != nil {
                                log.Debug("unmarshal message error: %v", err)
                                break
                        }
                        err = a.gate.Processor.Route(msg, a)
                        if err != nil {
                                log.Debug("route message error: %v", err)
                                break
                        }
                }
        }
}

func (a *agent) OnClose() {
        if a.gate.AgentChanRPC != nil {
                err := a.gate.AgentChanRPC.Call0("CloseAgent", a)
```

```
                if err != nil {
                        log.Error("chanrpc error: %v", err)
                }
        }
}

func (a *agent) WriteMsg(msg interface{}) {
        if a.gate.Processor != nil {
                data, err := a.gate.Processor.Marshal(msg)
                if err != nil {
                        log.Error("marshal message %v error: %v", reflect.TypeOf(msg), err)
                        return
                }
                err = a.conn.WriteMsg(data...)
                if err != nil {
                        log.Error("write message %v error: %v", reflect.TypeOf(msg), err)
                }
        }
}

//------------------------------新增-----------------------------------------
// Golang 语言社区
// cserli
// 2018 年 03 月 24 日
func (a *agent) PlaySendMessage(msg interface{}) {
        if a.gate.Processor != nil {
                data, err := a.gate.Processor.Marshal(msg)
                if err != nil {
                        log.Error("marshal message %v error: %v", reflect.TypeOf(msg), err)
                        return
                }
                err = a.conn.WriteMsg(data...)
                if err != nil {
                        log.Error("write message %v error: %v", reflect.TypeOf(msg), err)
                }
        }
}

//------------------------------修改-----------------------------------------

func (a *agent) LocalAddr() net.Addr {
        return a.conn.LocalAddr()
}
```

```
func (a *agent) RemoteAddr() net.Addr {
	return a.conn.RemoteAddr()
}

func (a *agent) Close() {
	a.conn.Close()
}

func (a *agent) Destroy() {
	a.conn.Destroy()
}

func (a *agent) UserData() interface{} {
	return a.userData
}

func (a *agent) SetUserData(data interface{}) {
	a.userData = data
}
```

6、network 讲解



Agent.go：

```
type Agent interface {
	Run()
	OnClose()
}
```

Conn.go:

```
type Conn interface {
        ReadMsg() ([]byte, error)
        WriteMsg(args ...[]byte) error
        LocalAddr() net.Addr
        RemoteAddr() net.Addr
        Close()
        Destroy()
}
```

7、课后作业　难度：★ ☆ ☆ ☆ ☆

根据本节课程的讲解：描述下 gate 在整个框架中的作用？

8、Go 语言微信公众平台及服务号

Golang 语言社区

Golang 技术社区

9、课程购买方式


INVITATION
Golang
Golang语言社区
向您推荐
LollipopGo社区开源架构实战项目
Golang语言社区
¥1200.00 ¥1500.00
课程概述 【本章内容】 Leaf框架设计思想，从原作者设计出发，Leaf框架底层leaf库分析， Le a...
网易云课堂
长按扫码 进入课程