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　Golang 语言社区-设计模式

## 第三节 单例模式

## 一、公众账号：


回复关键字：客服　　获取课程助教的微信（助教 MM）

## 二、单例模式讲解：

单例模式，是一种常用的软件设计模式，在它的核心结构中只包含一个被称为单例的特殊类。通过单例模式可以保证系统中一个类只有一个实例且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。

# 懒汉模式

懒汉模式是开源项目中使用最多的一种，最大的缺点是非线程安全的

```
type singleton struct {
}

// private
var instance *singleton

// public
func GetInstance() *singleton {
    if instance == nil {
        instance = &singleton{}     // not thread safe
    }
    return instance
}
```

# 带锁的单例模式

```
type singleton struct {
}

var instance *singleton
var mu sync.Mutex

func GetInstance() *singleton {
    mu.Lock()
    defer mu.Unlock()

    if instance == nil {
        instance = &singleton{}     // unnecessary locking if
instance already created
```

```
    }
    return instance
}
```

这里使用了 Go 的 sync.Mutex,其工作模型类似于 Linux 内核的 futex 对象，具体实现极其简单，性能也有保证
初始化时填入的 0 值将 mutex 设定在未锁定状态，同时保证时间开销最小
这一特性允许将 mutex 作为其它对象的子对象使用

# 带检查锁的单例模式

```
if instance == nil {     // <-- Not yet perfect. since it's no
t fully atomic
        mu.Lock()
        defer mu.Unlock()

        if instance == nil {
            instance = &singleton{}
        }
    }
    return instance
}
```

这是一个不错的方法，但是还并不是很完美。因为编译器优化没有检查实例存储状态。如果使用 sync/atomic 包的话就可以自动帮我们加载和设置标记。

```
import "sync"
import "sync/atomic"

var initialized uint32
...

func GetInstance() *singleton {

    if atomic.LoadUInt32(&initialized) == 1 {
        return instance
    }

    mu.Lock()
```

```
    defer mu.Unlock()

    if initialized == 0 {
         instance = &singleton{}
         atomic.StoreUint32(&initialized, 1)
    }

    return instance
}
```

## 比较好的一种方式 sync.Once

```
import (
    "sync"
)

type singleton struct {
}

var instance *singleton
var once sync.Once

func GetInstance() *singleton {
    once.Do(func() {
        instance = &singleton{}
    })
    return instance
}
```