qtask

提供稳定，可靠的任务管理机制，直到任务被成功处理。

实时任务：将任务存入 DB,并放入消息队列。业务系统订阅消息，处理逻辑，根据需要结束任务。未结束的任务超时后自动放入队列，继续处理。

延时任务：将任务存入 DB,超时后放入消息队列。业务系统订阅消息，处理逻辑，根据需要结束任务。未结束的任务超时后自动放入队列，继续处理。

特性:

√ 支持实时任务

√ 支持延时任务

√ 任务自动存储，防丢失，支持 mysql,oracle

√ 定时将任务放入队列

√ 过期任务定时清理

√ 一行代码安装任务表

√ 基于 hydra 构建

一、准备

创建任务表

1. 编译 `qtask`

```
~/work/bin$ go install github.com/micro-plat/qtask #mysql

或

~/work/bin$ go install -tags "oracle" github.com/micro-plat/qtask # oracle
```

```

```

2. 运行命令

`qtask [注册中心地址] [平台名称]` 即可将 `qtask` 需要的表创建到`/平台/var/db/db` 配置对应的数据库中。

```

```

```
~/work/bin$ qtask zk://192.168.0.109 mall #根据/mall/var/db/db创建数据库

或

~/work/bin$ qtask zk://192.168.0.109 mall mdb #根据/mall/var/db/mdb创建数据库
```

二、编码

1. 绑定服务

[概述] 1.定时清除任务 由于任务数据过多,需要每天凌晨定时清除已处理和处理失败并且达到删除期限的任务 2.定时扫描任务 定时扫描任务需要设置扫描任务的频率 主要处理达到执行期限或最大执行次数5次的任务,将任务处理失败,每次最多处理1000条 扫描需要执行的任务,每次最多处理1000条,发送消息队列到redis服务中

```
app.Initializing(func(c component.IContainer) error {
    qtask.Bind(app,10)  //每隔10秒将未完成的任务放入队列,app:*hydra.MicroApp
}
```

2. 创建任务

[概述] 任务主要分为实时任务和延时任务两种 实时任务主要处理需要立即执行的任务并支持事物,由于事物没有提交,发送消息队列可能查询不到,需要在事物提交之后,执行发送redis消息队列函数 延迟任务处理主要徐要延迟处理的任务

可选择传入参数

> qtask.WithDeadline 秒数，设置任务截止时间,默认为604800(7天) qtask.WithDeleteDeadline 秒数，设置任务删除截止时间,默认为0,执行成功,删除时间为当前时间,未执行成功,删除时间为当前时间+604800(7天)

```
// 业务逻辑

// 创建实时任务，将任务保存到数据库(状态为等待处理)并放入消息队列
// taskID:任务编号
// fcallback:发送当前任务消息队列到redis中函数
// err:返回错误
taskID, fcallback, err:=qtask.Create(c,"订单绑定任务",map[string]interface{}{
    "order_no":"8973097380045"
},3600,"GCR:ORDER:BIND")


//创建延时任务，将任务保存到数据库(状态为等待处理),超时后放入消息队列
// taskID:任务编号
// err:返回错误
taskID, err:=qtask.Delay(c,"订单绑定任务",map[string]interface{}{
```

```
    "order_no":"8973097380045"
},60,3600,"GCR:ORDER:BIND", qtask.WithDeadline(86400))
```

4. 处理 GCR:ORDER:BIND消息

```
func OrderBind(ctx *context.Context) (r interface{}) {
    //检查输入参数...

    //业务处理前调用，修改任务状态为处理中(超时前未调用qtask.Finish，任务会被重新放入队列)
    qtask.Processing(ctx,ctx.Request.GetInt64("task_id"))


    //处理业务逻辑...


    //业务处理成功，修改任务状态为完成(任务不再放入队列),并修改删除截止时间
    qtask.Finish(ctx,ctx.Request.GetInt64("task_id"))
}
```

三、其它

1. 自定义数据库名，队列名

```
qtask.Config("order_db","rds_queue") //配置数据库名，队列名
```

2. 使用不同的数据库

使用 mysql 数据库

```
 go install 或 go install -tags "mysql"
```

使用 oracle 数据库

```
 go install -tags "oracle"
```

完整示例