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

```
go install github.com/qtask/qtask #mysql
或
go install -tags "oci" github.com/qtask/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

```
qtask zk://192.168.0.109 mall #根据/mall/var/db/db创建数据库
或
qtask zk://192.168.0.109 mall mdb #根据/mall/var/db/mdb创建数据库
```

即创建到 /mall/var/db/mdb 配置的数据库中

二、编码

1. 绑定服务

```
app.Initializing(func(c component.IContainer) error {
    qtask.Bind(app,10,3) //每隔10秒将未完成的任务放入队列,删除3天前的任务,app:*hydra.MicroAppl
}
```

2. 创建任务

```
//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
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
qtask.Delay(c,"订单绑定任务",map[string]interface{}{
    "order_no":"8973097380045"
},3600,"GCR:ORDER:BIND",qtask.WithFirstTry(60), qtask.WithDeadline(86400))

qtask.WithFirstTry 设置首次放入队列时间
qtask.WithDeadline 设置任务截止时间
```

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)
    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 "oci"
```

完整示例