执行器(Worker)

一、简介：

worker既执行器，同时worker也是其包名，执行器的库代码在eago/src/task/worker中。在eago中，执行器是运行任务的最小单元。一个执行器内，可以注册和运行多个相同服务的不同任务。而某一个具体的服务（比如：eago-auth）最多只允许有1个执行器，但这1个执行器可以启动任多个执行器实例。

那么执行器提供什么功能呢？执行器可以提供让某一个具体服务的异步任务被task服务托管调度的功能。在你的服务中，如果有异步任务处理的需要，你应该：

1. 在task服务中创建一个任务，任务的codename应该遵循<你的服务名>.<任务名>；同时，在参数字段中输入这个服务的形参和类型；
2. 在你的服务中基于worker.TaskFunc实现你的任务处理方法；
3. 在你的服务中创建一个执行器接口实例，并且利用这个接口的RegTask方法将它注册到执行器；这里注意：任务名要和你在第一步创建的任务名相同；
4. 此后，你就可以通过TaskCli中的CallTask方法调用它了

二、如何使用执行器：

2.1. 创建自己的执行器

这里通过为test服务创建一个worker为例，以代码演示其过程：

test/worker/main.go

```
package main

import (
   "context"
   "eago/common/log"
   "eago/test/conf"
   "eago/task/worker"
   "fmt"
   "runtime"
)

func main() {
   // worker
   wk := worker.NewWorker(
      worker.EtcdAddresses(conf.Config.EtcdAddresses),
      worker.EtcdUsername(conf.Config.EtcdUsername),
      worker.EtcdPassword(conf.Config.EtcdPassword),
      // Woker
      worker.ServiceName("test"),
   )

   // test_workerTaskFunc
   wk.RegTask("test_worker", foo)

   // Worker
   if err := wk.Start(); err != nil {
      log.Error(err.Error())
   }
}
```

```
func init() {
   runtime.GOMAXPROCS(runtime.NumCPU())

   //
   err := log.InitLog(
      conf.Config.LogPath,
      conf.MODULAR_NAME,
      conf.Config.LogLevel,
   )
   if err != nil {
      fmt.Println("Failed to init logging, error:", err.Error())
      panic(err)
   }
}

// foo TaskFunc
func foo(ctx context.Context, param *Param) error {
   // paramlogtask.result
   defer param.Log.Info("test_worker ended.")

   param.Log.Info("Got test_worker call, and there is foo.")
   param.Log.Info(fmt.Sprintf("Your arugments is: %s", param.Arguments))

   // task.result""TaskFuncctx.Done
   // return niltask.result""
   // return error""
   // TaskFuncpanic""
   return nil
}
```

2.2. 其他补充

2.2.1. Log

使用param.Log的Info和Error方法，由这些方法打印的日志同时会被传送到Task.Result模块中。可通过任务模块的前端查看这些日志，如图：

2.2.2. 超时

如果任务调用者设置了任务超时时间，或者用户在任务结果页面上点击了结束任务，那么执行器会调用ctx的cancel方法，同时在task.result模块任务状态会被标记为“任务超时“。所以我们建议在您实现的TaskFunc中处理ctx，以便在出现上述情况时能正常结束您实现的TaskFunc。

任务超时的状态码为：-2

任务被手动结束的状态码为：-1

2.2.3. 错误

TaskFuncniltask.result“”0

TaskFuncerror“”-3

TaskFuncpanic“”-255

三、调用任务：

同样的，这里以调用test中test_worker任务需求，以代码为您演示如何实现

test/src/caller.go

```
package caller

import (
   "context"
   "eago/common/log"
   "eago/test/cli"
   task "eago/task/srv/proto"
   "fmt"
)

func main() {
   //
   req := &task.CallTaskReq{
      TaskCodename: "test.test_worker",
      Arguments:    "",
      Timeout:      0,
      Caller:       "eago-test",
   }

   //
   rsp, err := cli.TaskClient.CallTask(context.Background(), req)
   if err != nil {
      fmt.Println("Error: ", err.Error())
      return
   }
   log.Info("Call task done, task unique id: ", rsp.TaskUniqueId)
}
```

如果TaskClient.CallTask无法根据您提供的任务ID找到任务，任务状态会被标记为“调用错误“，状态码：-202

如果TaskClient.CallTask没有找到对应活跃的worker，任务状态会被标记为“找不到Worker“，状态码：-201

如果TaskClient.CallTask找到活跃的worker但worker没有对应的任务，任务状态会被标记为“找不到任务“，状态码：-200

*注：状态代码越小，错误越严重