构建六大服务器

一、构建 api server

```
package main

import (
    "github.com/micro-plat/hydra"
    "github.com/micro-plat/hydra/hydra/servers/http"
)

func main() {

    //创建app
        app := hydra.NewApp(
            hydra.WithServerTypes(http.API),
    )

    //注册服务
    app.API("/hello", hello)

    //启动app
    app.Start()
}
func hello(ctx hydra.IContext) interface{} {
        return "success"
}
```

二、构建mqc server

```
package main

import (
    "github.com/micro-plat/hydra"
    "github.com/micro-plat/hydra/hydra/servers/mqc"
)

func main() {

    //创建app
        app := hydra.NewApp(
            hydra.WithServerTypes(mqc.MQC),
    )

    //注册服务
    app.MQC("hello",hello,"queue-name")//接收“queue-name”的消息，并执行/hello"服务

    //启动app
    app.Start()
}
func hello(ctx hydra.IContext) interface{} {
        return "success"
}
```

三、构建cron server

```
package main

import (
    "github.com/micro-plat/hydra"

     "github.com/micro-plat/hydra/hydra/servers/cron"
)

func main() {

    //创建app
        app := hydra.NewApp(
            hydra.WithServerTypes(cron.CRON),
    )

    //注册服务
    app.CRON("/hello",hello,"@every 5s") //每隔5秒执行一次"/hello"服务


    //启动app
    app.Start()
}
func hello(ctx hydra.IContext) interface{} {
        return "success"
}
```

四、构建rpc server

```
package main

import (
    "github.com/micro-plat/hydra"

     "github.com/micro-plat/hydra/hydra/servers/rpc"
)

func main() {

    //创建app
        app := hydra.NewApp(
            hydra.WithServerTypes(rpc.RPC),
    )

    //注册服务
    app.RPC("/hello",hello)


    //启动app
    app.Start()
}
func hello(ctx hydra.IContext) interface{} {
        return "success"
}
```

五、构建ws server

```go
package main

import (
    "github.com/micro-plat/hydra"

     "github.com/micro-plat/hydra/hydra/servers/http"
)

func main() {

    //创建app
        app := hydra.NewApp(
            hydra.WithServerTypes(http.WS),
    )

    //注册服务
    app.WS("/hello",hello)


    //启动app
    app.Start()
}
func hello(ctx hydra.IContext) interface{} {
        return "success"
}
```