Day11 微服务入门之编写HelloWorld

1 打卡任务

作业：

1. 在provider服务中增加一个greeting方法，接受POST请求，请求参数为request body中传递的Person对象，包含name和gender两个字段。name为String类型；gender为枚举Gender类型，有MALE/FEMALE两个值。返回值为GreetingResponse类型，包含msg和timestamp两个字段，msg为根据gender不同生成的问候语 Hello, Mr.xxx (MALE) / Hello, Ms.xxx (FEMALE)，timestamp为java.util.Date类型的时间戳。
2. 在consumer服务中增加一个greeting方法，接收外部请求的触发，以调用provider服务的greeting方法，并返回provider的应答消息。
    i. 打卡：
3. 调用provider服务的greeting方法成功，并截图
4. 调用consumer服务的greeting方法成功，并截图

开发基于 Day11 的 demo

2 准备工作

1. 本地成功运行 Day11 的provider、consumer demo工程项目。

3 在 provider 服务中开发 greeting 方法

1. 创建 GreetingResponse 和 Person，以及MEAL和FEMEAL的声明

```
const Male string = "MALE"
const Female string = "FEMALE"

type GreetingResponse struct {
    Msg       string        `json:"msg"`
    Timestamp time.Duration `json:"timestamp"`
}
type Person struct {
    Name   string `json:"name"`
    Gender string `json:"gender"`
}
```

2，为 service 创建 Greeting 方法

```
func (*Service) Greeting(ctx *restful.Context) {
    param := Person{}
    ctx.ReadEntity(&param)
    r := GreetingResponse{}
    if param.Gender == Male {
        r.Msg = fmt.Sprintf( format: "Hello , Mr.%s", param.Name)
    } else if param.Gender == Female {
        r.Msg = fmt.Sprintf( format: "Hello , Ms.%s", param.Name)
    }
    r.Timestamp = time.Duration(time.Now().UnixNano() / 1e6)
    ctx.WriteJSON(r, contentType: "application/json")
}

// URLPatterns
func (*Service) URLPatterns() []restful.Route {
    return []restful.Route{
        {Method: http.MethodPost, Path: "/greeting", ResourceFuncName: "Greeting"},
    }
}
```

3，启动 provider 服务，调用它的 greeting 接口，可以看到正常返回结果。

4 在 consumer 服务中开发 greeting 方法

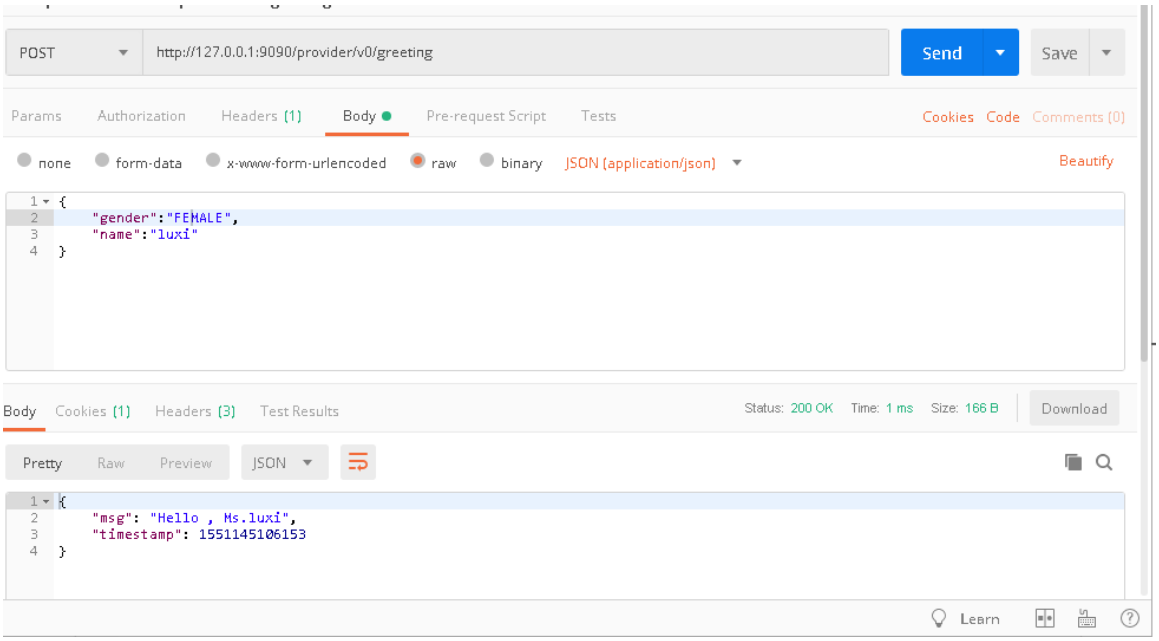
1 ,为 consumer 创建 Greeting 方法 ,在方法中我们需要将请求中 body 参数传递至 provider

```go
func (*Client) Greeting(ctx *restful.Context) {

    req, err := rest.NewRequest(http.MethodPost, urlStr: "cse://server-demo/greeting", body: nil)
    if err != nil {
        ctx.WriteError(http.StatusInternalServerError, err)
        return
    }
    req.Body = ctx.ReadRequest().Body
    resp, err := core.NewRestInvoker().ContextDo(context.TODO(), req)
    if err != nil {
        ctx.WriteError(http.StatusInternalServerError, err)
        return
    }
    ctx.Write(httputil.ReadBody(resp))
}

// URLPatterns
func (*Client) URLPatterns() []restful.Route {
    return []restful.Route{
        {Method: http.MethodPost, Path: "/greeting", ResourceFuncName: "Greeting"},
    }
}
```

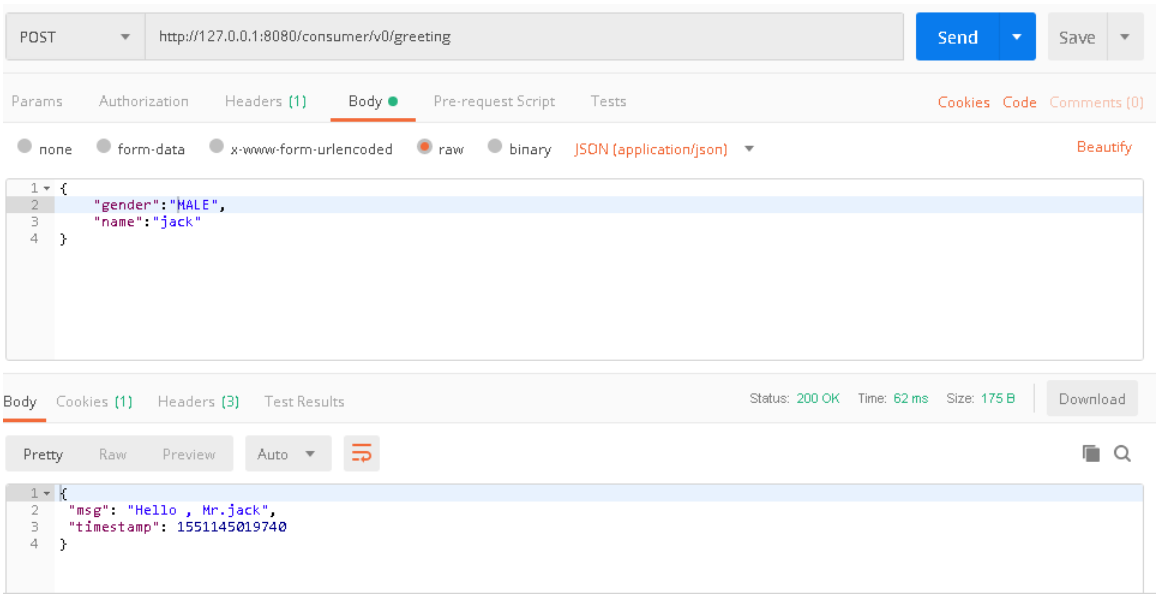
2，启动 consumer 服务，调用它的 greeting 接口，可以看到正常返回结果。

5 打卡截图

1，调用provider服务的结果如下：



2，调用consumer服务结果如下



6 总结

1， 在该demo中，我们仅介绍了CseGoSdk的rest协议的开发， 除了支持rest协议之外同时也支持grpc协议，如果需要了解更多相关文档请参考，CseGoSdk文档。

2， 在使用过程中如遇到问题，欢迎在华为云社区进行提问，你也可以到开源代码库提issue