<https://github.com/emicklei/go-restful>

# 描述

REST要求开发人员明确使用HTTP方法，并以与协议定义一致的方式。 这种基本的REST设计原则在创建，读取，更新和删除（CRUD）操作和HTTP方法之间建立一对一的映射。

根据这个映射：

GET = 取回资源的描述

POST = 服务端接受客户端的参数并创建资源.

PUT = 创建如果您正在发送指定资源（URI）的完整内容.

PUT = 如果要更新指定资源的完整内容，请更新.

DELETE = 如果要求服务器删除资源，请删除

PATCH = 更新资源的部分内容

OPTIONS = 获取有关请求URI的通信选项的信息

# 实例

ws := new(restful.WebService)

ws.Path("/users").

Consumes(restful.MIME\_XML, restful.MIME\_JSON).

Produces(restful.MIME\_JSON, restful.MIME\_XML)

ws.Route(ws.GET("/{user-id}").To(u.findUser).

Doc("get a user").

Param(ws.PathParameter("user-id", "identifier of the user").DataType("string")).

Writes(User{}))

...

func (u UserResource) findUser(request \*restful.Request, response \*restful.Response) {

id := request.PathParameter("user-id")

...

}

# 特点

1. 路由请求→功能映射与路径参数（例如{id}）支持
2. 可配置路由器
   1. （默认）快速路由算法，允许URL路径中的静态元素，正则表达式和动态参数（例如/ meetings / {id}或/ static / {subpath：\*}
   2. 使用（但不接受）正则表达式实现JSR311后的路由算法
3. 请求API从JSON / XML读取结构体并访问参数（路径，查询，标题）
4. 用于将结构体编写为JSON / XML并设置标题的Response API
5. 使用EntityReaderWriter注册的可定制编码
6. 用于拦截请求的过滤器→服务或路由级别的响应流程
7. 使用属性的请求范围变量
8. Web服务的容器在不同的HTTP端点上
9. 请求和响应有效内容的内容编码（gzip，deflate）
10. 在OPTIONS上自动响应（使用过滤器）
11. 自动CORS请求处理（使用过滤器）
12. API declaration for Swagger UI (go-restful-openapi, see go-restful-swagger12)
13. 紧急恢复生成HTTP 500，可使用RecoverHandler（...）进行自定义
14. 路由错误产生HTTP 404/405/406/415错误，可使用ServiceErrorHandler（...）进行自定义
15. 可配置（跟踪）日志记录
16. 使用CompressorProvider注册可定制gzip /放大读者和作者