

随堂练习

本次练习是在REST服务集成的初期讲解基础上，为了加深对REST服务的直观理解，同时也为了接下来的内容，而进行的认识、巩固练习。

练习的内容包括：

1. 对我们服务器中搭建的（You type it, we post it）网站进行Web API操作；
2. 以我们的教辅材料中的Go原始代码为基础，进一步加深对Web API实现的理解。

练习一

1、目的

该练习将对我们服务器中搭建的（You type it, we post it）网站进行Web API操作，以此了解各种http方法的操作，并与网页操作就行对比，从而加深对http协议的理解。

2、实验材料

- 1) 网站: <http://119.36.242.188:9980/type> .
- 2) curl工具。请根据自己电脑的OS，自行下载各自版本的curl软件工具

Windows用户需要特别注意。Win10以上的系统内置了curl工具。这本是好事。但是，由于内置的curl有以下问题：

- 不能识别 `-X PUT`、`-X DELETE` 等http方法，即，只能使用 `GET` 和 `POST` 两种http方法
- 在解析一些参数时，自作主张地加入了一些额外字符，如：“\”符号等
- 与linux等标准的curl相比，有些其它使用上的差别

因此，不建议使用Windows系统原生的curl命令。可以自行下载curl-openssl等第三方的工具，取代原生的curl。

- 3) template文件。可以按照教辅材料中type.zip中包含的样本，自行创建。该zip包中，包含网站的原始代码。故，学生也可以在自己的电脑上，自行运行web service，以取代远程网站的访问。
- 4) node.js。由于网站采用Node.js编程，如果需要自行运行web service，需要安装Node.js环境。

3、练习内容

1) 网站操作

访问<http://119.36.242.188:9980/type>，对其进行标头（Header）解析、内容输入，观察结果，结合课堂讲解的理论，加深理解。

2) Web API操作

使用curl作为工具，通过命令行方式，对服务端API进行操作：

- 使用 /GET 方法，获取 /api 的内容
- 使用 /POST 方法，写入自己的内容
- 使用 /PUT 方法，对某个ID指向的内容进行修改
- 使用 /DELETE 方法，将某个ID指向的内容删除

在进行上述内容操作的同时，对照代码，理解其工作的原理。

3) 源代码

源代码在教辅材料的type.zip中。

- 阅读并理解
- 运行主程序app.js: `node app.js`
- 尝试修改源代码，实现自己的API目的

练习二

1、目的

以我们的教辅材料中的Go原始代码为基础，进行各种操作，进一步加深对Web API实现的理解。

2、实验材料

1) 我们提供的Web服务：

- <http://119.36.242.188:9980/goAPI0/list>
- <http://119.36.242.188:9980/goAPI1/list>
- <http://119.36.242.188:9980/goAPI2/list>
- <http://119.36.242.188:9980/goAPI3/list>

除上述服务（/list）外，每个端口还包括：/add、/update 路由服务。其中包含/GET /POST /PUT方法。

2) 原始代码。代码在教辅材料的 /golang目录中。

▼	golang	•
▼	pkg	•
▼	mod	•
	 gopkg.in.zip	385 KB
▼	REST_exc	•
>	config	•
	 server3.go	8 KB
	 server2.go	7 KB
	 server1.go	4 KB
	 server0.go	3 KB
	 logfile.txt	6 KB

其中，REST_exc/目录中为原始go代码：

- server0.go：v1.0版代码；
- server1.go：v1.1版代码；
- server2.go：v1.2版代码；
- server3.go：v1.3版代码；




从1.0到1.3，不同版本的代码，体现了不同的API特点，需要自行理解。

pkg/mod/目录中，包含编译go代码所需的第三方库：gopkg.in，用于解析标准的yaml文件格式。学生可根据其所使用的操作系统、安装go的环境不同，自行拷贝到相应的目录，以支持代码的编译。

config/ 目录中，包含auth.yaml（user/token组合验证信息）、content.yaml（读、写的资源数据）两个文件。

3)、go语言环境

根据自己所使用的OS，自行安装相应的go环境。教辅材料中，已经存在3种不同的安装包：

	go1.21.3.windows-amd64.msi	61.3 MB
	go1.21.3.linux-amd64.tar.gz	66.6 MB
	go1.21.3.darwin-arm64.pkg	65.9 MB

它们是当前最新的go版本。其中，.msi是Windows版本，.gz是linux版本，而.pkg是MacOS版本（Apple M系列芯片）。如需要自行下载，也可以到：<http://golang.google.cn/dl> 下载。

特别提示：安装完毕后，需要配置GOPATH环境变量（根据OS情况，配置方法有所不同）。这样，go才能找到公共运行包所处的位置。有的OS还需要自行配置环境指向：go的bin目录。

3、练习内容

1) Web API操作 (curl)

对我们提供的远程公共API进行 /list /add /update路由操作。使用curl进行。

数据源是一些人员的列表，每个人包含的信息：编号（API自动生成）、姓名、性别、工作。

如，获取API的描述信息，在1.0版API中：

```
1 curl http://119.36.242.188:9980/goAPI0
```

得到的结果为：v1.0版，原始版本，包含：/list /add /update三个路由。

如，对于人员列表操作，在1.0版API中：

```
1 curl -v http://119.36.242.188:9980/goAPI0/list # -v等同于--verbose，给出详细信息
```

v1.0版本无安全认证需要。

如，对于人员添加，在1.1版本中：

```
1 curl -v -H "X-User-ID: user1" -H "Authorization: Bearer st%ss4{iYb0;h<tt" -X POST -d "name={姓名}&gender={性别}&job={工作性质}" http://119.36.242.188:9980/goAPI1/add
```

v1.1，以及以上版本，均需要安全认证。

如，对于人员更新，在1.3版本中：

```
1 curl -v -H "X-User-ID: user2" -H "Authorization: Bearer ^F25fsp{;dT7<uwY" -X PUT -d "id={id}&name={姓名}&gender={性别}&job={工作性质}" http://119.36.242.188:9980/goAPI3/update #注意，与POST相比，多了id参数，即需要更新的id
```

安全认证信息不能更改，它们保存在服务器端。如果部署在你的本地机器，可以自行修改。该信息保存于 config/auth.yaml中。

上述例子中，{内容}可以取代（取代后，将大括弧去掉）。如需要空格，使用%20代替。

注意版本号、端口号的一致性。

2) 浏览器操作 (建议：chrome)

对于上述API访问，使用浏览器，输入对应的url进行访问。

如：对于v1.0版，获取信息列表：

```
1 http://119.36.242.188:9980/goAPI0/list
```

如：对于v1.0版，增加信息内容：

```
1 http://119.36.242.188:9980/goAPI0/add?name=张三&gender=male&job=软件测试工程师
```

对于v1.1以及以上版本，访问方式不变。但是，由于加入了安全机制，需要通过浏览器内置的开发者工具，修改请求的Header信息，即，手工输入user/token组合，进行访问。对于Chrome浏览器，可以安装ModHeader插件进行。

如果不能通过安全验证，所有请求都返回 `Unauthorized` 信息，返回状态码为：401.

3) 观察

- 观察不同的http方法，其go代码实现的方式
- 观察不同版本的go代码，它们的差异在哪里？
- 观察浏览器操作与curl操作的不同
- 观察curl操作返回的标头信息（尤其是状态码信息）
- 每次操作（无论是浏览器操作，还是curl操作），在v1.2及以上版本，均有本地log记录。如果是在本地运行go代码，可以观察logfile.txt文件
- 观察v1.2与v1.3中，对于/add /update的操作，它们采用的http方法有哪些异和同？

4) 自行编译go代码

自行编译go代码，并部署在本地机器中。

```
1 | go build -o {你的目标应用名字} {你的go源码}
```

也可以不经编译，直接运行：

```
1 | go run {你的go源代码}
```

你可以按照自己的需要，修改原始代码，实现自己的目标。如：

- 增加 /delete/{id} 路由，实现 /DELETE方法，删除某个id的内容
- 修改路由对应的http方法。如：使用 /POST 对某个id的内容进行修改
- 改善幂等性
- 改善API的使用体验
- ...