5. REST

- SOAP与REST对比

SOAP作为一种古老的Web服务技术，短期内还不会退出历史舞台。但其仅支持XML, 和其较为复杂的解析操作。

与SOAP相比， REST 使用了标准 HTTP ，因此其创建客户端，开发 API，编写文档都会更加简单

5. REST

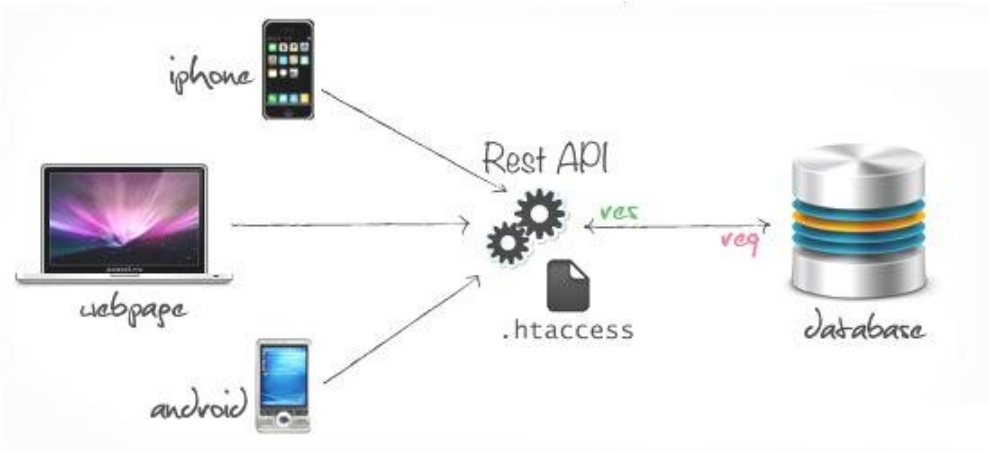
- REST(Representational State Transfer)
  1. 是Roy Thomas Fielding博士于2000年在他的博士论文中提出来的一种万维网软件架构风格，目的是便于不同软件/程序在网络（例如互联网）中互相传递信息。
  2. 资源是由URI来指定。对资源的操作包括获取、创建、修改和删除资源，这些操作正好对应HTTP协议提供的GET、POST、PUT和DELETE方法。
  3. 用 HTTP Status Code传递Server的状态信息。比如最常用的 200 表示成功，500 表示Server内部错误等。

5. REST

- REST(Representational State Transfer)

用HTTP协议里的动词来实现资源的添加，修改，删除等操作。即通过HTTP动词来实现资源的状态扭转：

1. GET　用来获取资源
2. POST　用来新建资源
3. PUT　用来更新资源
4. DELETE　用来删除资源。



5. REST

- API访问

例子: 需要获取Github中user为octocat的信息， 可以通过Github提供的API， 通过访问该API获取相应的信息。方式有如下：

1. 在浏览器中输入https://api.github.com/users/octocat
2. 在命令行下使用cURL工具

   curl https://api.github.com/users/octocat

5. REST

- API访问


```
{
 "login": "octocat",
 "id": 583231,
 "avatar_url": "https://avatars3.githubusercontent.com/u/583231?v=4",
 "gravatar_id": "",
 "url": "https://api.github.com/users/octocat",
 "html_url": "https://github.com/octocat",
 "followers_url": "https://api.github.com/users/octocat/followers",
 "following_url": "https://api.github.com/users/octocat/following{/other_user}",
 "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
 "starred_url": "https://api.github.com/users/octocat/starred{/owner}{/repo}",
 "subscriptions_url": "https://api.github.com/users/octocat/subscriptions",
 "organizations_url": "https://api.github.com/users/octocat/orgs",
 "repos_url": "https://api.github.com/users/octocat/repos",
 "events_url": "https://api.github.com/users/octocat/events{/privacy}",
 "received_events_url": "https://api.github.com/users/octocat/received_events",
 "type": "User",
 "site_admin": false,
 "name": "The Octocat",
 "company": "GitHub",
 "blog": "http://www.github.com/blog",
 "location": "San Francisco",
 "email": null,
 "hireable": null,
 "bio": null,
 "public_repos": 7,
 "public_gists": 8,
 "followers": 2014,
 "following": 5,
 "created_at": "2011-01-25T18:44:36Z",
 "updated_at": "2017-11-23T04:10:31Z"
}
```


GET请求返回结果

5. REST

- API访问

POST 请求： 利用POST请求在Github账号上创建一个gist， 发送携带JSON格式的数据在HTTPBody中, 可以在该账号上根据相应的数据创建相应的Gist.

```
{
  "description": "the description for this gist",
  "public": true,
  "files": {
    "file1.txt": {
      "content": "String file contents"
    }
  }
}
```

5. REST

- API访问

PATCH 请求： 利用POST请求在Github账号上修改一个gist， 发送携带JSON格式的数据在HTTPBody中, 可以在该账号上根据携带的数据修改相应的Gist.

```
{
  "description": "the description for this gist",
  "files": {
    "file1.txt": {
      "content": "updated file contents"
    },
    "old_name.txt": {
      "filename": "new_name.txt",
      "content": "modified contents"
    },
    "new_file.txt": {
      "content": "a new file"
    },
    "delete_this_file.txt": null
  }
}
```

5. REST

- API访问

DELETE： 利用DELETE请求在Github账号上删除对应id的gist

如： 向服务端对应API发送DELETE请求

DELETE /gists/:id

服务端Response

Status: 204 No Content