REST API Design

目录

[1：首先要遵循 http 1.1 REF 文档设计规则 2](#_Toc486499496)

[1.1：Option类型 2](#_Toc486499497)

[1.2：状态码 2](#_Toc486499498)

[1.3 ACCEPT 3](#_Toc486499499)

[1.4 ALLow 3](#_Toc486499500)

[1.5 缓存控制” 3](#_Toc486499501)

[1.6 连接 4](#_Toc486499502)

[1.7 内容 4](#_Toc486499503)

[1.8 日期 4](#_Toc486499504)

[1.9 Host 4](#_Toc486499505)

[2.0 ETag 与缓存之间的关系 4](#_Toc486499506)

[3.0 设计api的时候，考虑到幂等性，是非常重要的 4](#_Toc486499507)

[4.0 返回数据时遵守RFC的status Code 5](#_Toc486499508)

[5.0 安全性 5](#_Toc486499509)

[5.1 请求数据的验证 5](#_Toc486499510)

[5.2 数据完整性验证 5](#_Toc486499511)

[5.3 访问控制 5](#_Toc486499512)

[5.4 Https 6](#_Toc486499513)

[5.5 Rate limiting 6](#_Toc486499514)

[5.6 metrics 6](#_Toc486499515)

[5.7 docs 6](#_Toc486499516)

[5.8 hooks/event propogation 6](#_Toc486499517)

[5.9 REST API framework/library 7](#_Toc486499518)

[6.0 版本性控制 7](#_Toc486499519)

[7.0 API的根URL 7](#_Toc486499520)

[8.0过滤器 7](#_Toc486499521)

[9.0 错误提示信息 7](#_Toc486499522)

# 1：首先要遵循 http 1.1 REF 文档设计规则

* 参考文档：<https://tools.ietf.org/html/rfc2616>
* 参考页面：<http://www.cnblogs.com/Joans/p/3956490.html>
* 参考页面：<http://kb.cnblogs.com/page/521718/>

## 1.1：Option类型

* Get检索信息
* HEAD 用于测试超文本链接的有效性，可访问性和最近的修改（不得响应中返回消息体）
* POST 提交资源（200 OK）（204 无内容）201（创建）
* PUT 创建修改资源 （200 OK）（204 无内容）201（创建）501（未实施）
* DELETE 服务器删除资源 （200 OK）（204 无内容）

## 1.2：状态码

* 1XX 临时响应
* 100 继续 客户应该继续请求
* 2XX 成功
* 200 OK 201 创建 202 接受 请求被接受，但处理过程未完成
* 203 非授权信息 204 无内容 205 重置内容 206 部分内容响应
* 3XX 重定向
* 304 未修改 305 使用代理 307 临时重定向
* 4XX 客户端错误
* 400 错误请求，由于格式错误，服务端无法理解 401 未经授权 请求需要用户认证 403 服务器了解该请求，但拒绝履行该请求 404 未找到 405 不允许方法 406 不可接受 407 需要代理验证 408 请求超时 409 冲突 410 资源不可用 411 需要长度 413 请求实体太大 414 请求的url太长 415 不支持的介质类型 416 请求的范围不满意 412  PreCondition Fail
* 5XX 内部服务器错误
* 501 未实施 502 坏网关 503 服务不可用 504 网关超时 505 不支持HTTP版本

## 1.3 ACCEPT

* Accept ，Accept-Charset 字符集 Accept-Encoding 接受的字符集（compress，gzip） Accept-Language 接受的语言 Accept-Ranges 允许服务器接受的资源范围

## 1.4 ALLow

* Allow=“Get，HEAD，PUT

## 1.5 缓存控制”

* Cache-Control：”no-cache”
* Paragma:no-cache 1.0 协议
* Expires=”Expires”
* Max-age=”1”

## 1.6 连接

* Connection=”Connection”

## 1.7 内容

* Content-Encoding=“gzip”
* Content-Language=en
* Content-Length:456
* Content-Range=”Content-Range” Content-Location=”url”Content-Type=”Content-Type”

## 1.8 日期

* Date=”Date”

## 1.9 Host

* Host=“host：port”

# 2.0 ETag 与缓存之间的关系

* 参考文档 <http://www.cnblogs.com/tyb1222/archive/2011/12/24/2300246.html>
  + - * <http://www.maixj.net/wz/http-etag-1191>

# 3.0 设计api的时候，考虑到幂等性，是非常重要的

* 参考文档<http://www.cnblogs.com/weidagang2046/archive/2011/06/04/2063696.html>
* Get /DELETE/PUT 具有幂等性
* POST/PATCH 不具有幂等性
* Accept 服务器返回什么 返回错误类型 status code 406 not acceptable
* If-Modified-Since/If-None-Match 304 not modified
* If-Match 某个资源做PUT/PATCH/DELETE操作 否则返回412 precondition failed

# 4.0 返回数据时遵守RFC的status Code

* 参考状态码[1.2：状态码](#_1.2：状态码)

# 5.0 安全性

* 一致性（integrity），机密性（confidentiality）和可用性（availability）

## 5.1 请求数据的验证

* Request headers是否合法 出现了某些不该有的头，或者某些必须包含的头没有出现或者内容不合法，根据其错误类型一律返回4xx，API需要某个特殊的私有头（e.g. X-Request-ID），可以防止crawler的请求，节省服务器的开销
* Request URI和Request body是否合法 如果请求带有了不该有的数据，或者某些必须包含的数据没有出现或内容不合法，一律返回4xx

## 5.2 数据完整性验证

* 保证要修改的数据和服务器里的数据是一致的 —— 这是通过Etag来完成，服务器检查客户端提供的Etag是否和服务器同一资源的Etag相同，如果相同，才进行修改，否则返回412 precondition failed。

## 5.3 访问控制

* REST API需要清晰定义哪些操作能够公开访问，哪些操作需要授权访问
* 如果对REST API的安全性要求比较高，那么，所有的API的所有操作均需得到授权。
* 在HTTP协议之上处理授权有很多方法，如HTTP BASIC Auth，OAuth，HMAC Auth等
* HMAC 通过 secretKey ，对head加密，head里面有Timespan，来控制repaly攻击
* Replay攻击参考文档：<http://blog.csdn.net/koastal/article/details/53456696>

## 5.4 Https

* HMAC Auth尽管在保证请求的一致性上非常安全，可以用于鉴别请求是否由合法的请求者发起，但请求的数据和服务器返回的响应都是明文传输，对某些要求比较高的API来说，安全级别还不够。这时候，需要部署HTTPS。在其之上再加一层屏障。

## 5.5 Rate limiting

* 参考文档：<http://developer.51cto.com/art/201510/495062.htm>
* 参考文档：<http://www.cnblogs.com/exceptioneye/p/4783904.html>
* 漏桶策略，令牌策略

## 5.6 metrics

* 服务器应该收集每个请求的访问时间，到达时间，处理时间，于了解API的性能和客户端的访问分布，以便更好地优化性能和应对突发请求。

## 5.7 docs

* 丰富的接口文档
* Swagger：参考文档<http://blog.csdn.net/wangnan9279/article/details/44541665>
* <http://www.cnblogs.com/whitewolf/p/4686154.html>
* Web api 自动生成工具
* 参考文档：<https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/creating-api-help-pages>

## 5.8 hooks/event propogation

* 构建了webapi接口可以通过过kafka或者rabbitMQ向外界暴露某个消

## 5.9 REST API framework/library

* Django REST framework
* 参考文档：<http://www.cnblogs.com/xiaojikuaipao/p/6009882.html>
* .net webapi: <http://www.infoq.com/cn/news/2012/09/rest-web-api/>

# 6.0 版本性控制

* 为了确保服务器程序能够进化升级，同时能够让使用方感到满意，你需要在引入新版本API的同时继续让旧版本的API正常工作
* 文档对于API来说太重要了，没有好的文档必然没有好的API，因此API和它的文档一定同步修改，甚至要先修改文档
* 过一段时间之后，你可以告诉调用方不建议（deprecate）使用旧版本[逍遥子笔记：就像[Java](http://lib.csdn.net/base/java)里面的depredated注释一样，用于告诉使用方，我不建议你使用它了，过一段时间之后我可能就不支持它了]。不建议使用一个API并不意味着马上就要关闭它或者降低它的服务质量，而是告诉你的API使用人员他们需要版本升级，旧的版本将在未来一段时间之后被停止服务。

# 7.0 API的根URL

* 重要的是你API的URL根应该尽可能简单，一个又长又复杂的URL看起来就吓人，它很可能就把这些第三方开发人员吓跑了。

# 8.0过滤器

* 当用户请求获取一组对象列表时，你就需要对结果进行过滤并返回一组严格符合用户要求的对象
* 不要在接口中添加默认的约束条件

# 9.0 错误提示信息

* 不要把服务内部的错误或者问题暴露给第三方开发人员
* 提示信息简洁清晰方便管理