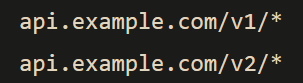
# RESTFul API

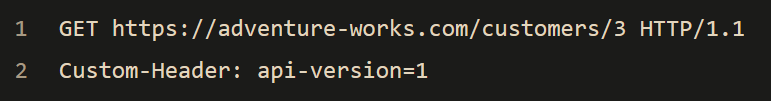
1. RESTFul API
   1. 是一种成熟的API规范，用于约束前后端交互的接口设计
2. 设计规范
   1. 协议
      1. 通信规范，目前应使用https
   2. 域名
      1. API的访问域名应当保持简洁
      2. api.example.com/\* （子域名下）
      3. example.com/api/\* （主域名下）
      4. 考虑到API的扩展性，最好放置于子域名下
   3. 路径
      1. 路径命名必须小写
      2. 资源命名是名词，且为复数
      3. 连字符使用" - "而非" \_ "，由于下划线可能被遮挡，UI字体与开发字体往往不同
      4. 例子
         1. 正例
            1. [api.example.com/zoos](https://link.juejin.cn/?target=https%3A%2F%2Fapi.example.com%2Fzoos)
            2. [api.example.com/zoos/animal](http://api.example.com/zoos/animal)
         2. 反例
            1. [api.example.com/getUser](https://link.juejin.cn/?target=https%3A%2F%2Fapi.example.com%2FgetUser)
            2. [api.example.com/addUser](http://api.example.com/addUser)
   4. HTTP动词
      1. 在http协议中，请求报文的类型，即动词，有以下几种
         1. GET（SELECT）：从服务器取出资源（一项或多项）
         2. POST（CREATE）：在服务器新建一个资源
         3. PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）
         4. PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）
         5. DELETE（DELETE）：从服务器删除资源
      2. 例子



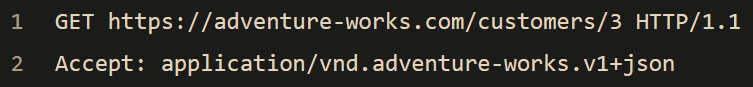
* 1. 信息过滤
     1. 有时后端返回的数据量过大，前端需要提供一些参数用于过滤，比如分页展示，排序，部分加载等
     2. 例子
        1. ?limit=10：指定返回记录的数量
        2. ?offset=10：指定返回记录的开始位置。
        3. ?page=2&per\_page=100：指定第几页，以及每页的记录数。
        4. ?sortby=name&order=asc：指定返回结果按照哪个属性排序，以及排序顺序。
        5. ?animal\_type\_id=1：指定筛选条件
  2. HATEOAS（*Hypermedia As The Engine Of Application State*）
     1. 这一概念用于：http后端返回的报文中，携带有其他后续API的链接，可以让客户方便地进行下一步操作
     2. 例子
        1. github的全部接口列表：[api.github.com](https://api.github.com/)
        2. UraraO的github用户接口调用的返回值，其中包含了其他API的地址：[api.github.com/users/UraraO](http://api.github.com/users/UraraO)
  3. 版本控制
     1. API的开发需要向前兼容，不能因为API版本更新导致过去的程序无法正常请求服务器
     2. 版本控制形式
        1. 1. URL中添加版本：此方式灵活性不强，维护成本较高，且不便于设计HATEOAS



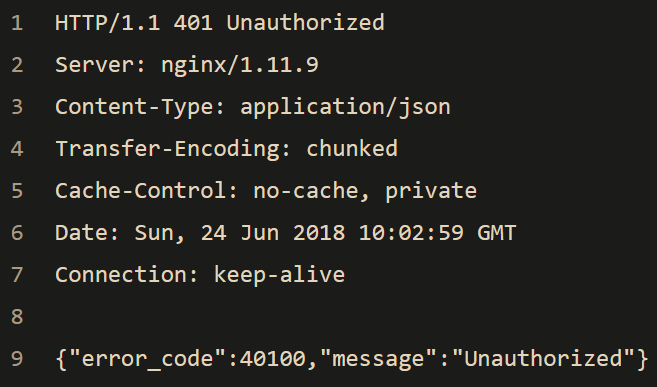
* + - 1. 2. 字符串查询：此方式使得同一资源永远从同一URI查询，确保了API的稳定性；但是也不便于设计HATEOAS
         1. 例如：<https://adventure-works.com/customers/3?version=2>
      2. 3. 自定义请求标头：请求头部中直接带版本信息，将版本控制从URL中转移到报文中



* + - 1. 4. 使用Accept标头：在标头中启用Accept字段，其中附带版本要求和其他形式要求，版本控制自然且适合HATEOAS



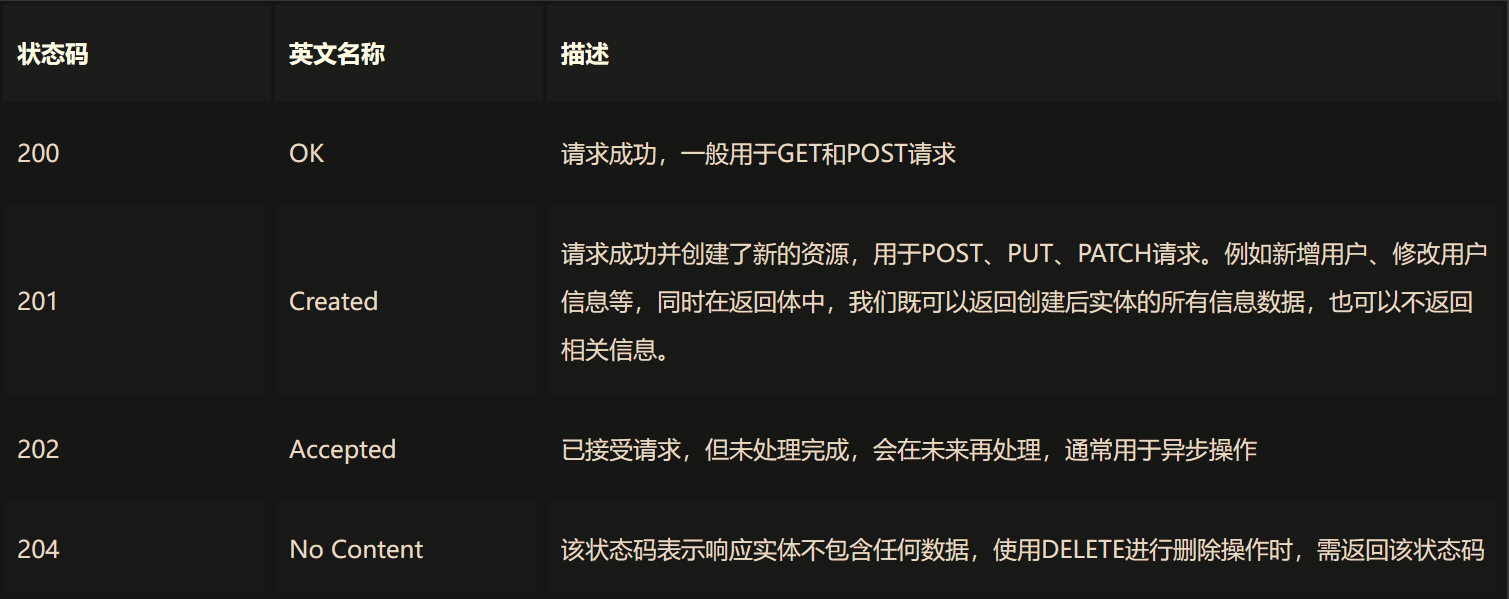
* 1. 服务器响应
     1. 服务器根据消息处理情况，产生响应报文，返回给客户端
     2. API设计需要遵守http设计规范：
        1. 返回http报文需要选择合适的状态码
        2. 包含合适的消息体
        3. 如果出现错误，需要包含错误信息



* 1. http状态码
     1. 状态码是http报文中的重要部分，由 3 位十进制数组成，第一位表示状态类型
     2. 分类



* + 1. 2xx 成功



* + 1. 3xx 重定向：其中303返回一个新API，由客户决定如何操作；API级别不需要使用3xx
    2. 4xx 客户端错误



* + 1. 5xx 服务端错误

