API v2

本文档是设备管理系统第二版接口开发的设计原则，我们还在不断完善它。

Introduction

设备管理系统是一套可管理设备的短租和长租业务的系统，具有设备信息录入、设备租赁、订单管理、消息提醒等功能。我们提供一个 REST 风格的 API，它的设计参考了best-practices-for-a-pragmatic-restful-api。

我们的 API 使用面向资源的 URLs，同时较好地利用了 HTTP 本身的一些特征，如 HTTP 动词、HTTP 响应状态码。所有请求和响应的消息体都是 JSON 格式，包括错误响应。任何 HTTP 客户端（浏览器也不例外）都可以用来与 API 进行通信。

我们认为 API 是开发者的用户界面，因此，我们也尽最大努力保证 API 能够方便地通过浏览器直接就可以访问。

Authentication

内部使用的接口，不需要认证即可调用。

Requests

API 的基础地址是 http://10.20.17.211:8008/v2，地址中的 v2 标识当前接口的版本，调用时请替换成实际的版本。

JSON Bodies

所有 POST, PUT, PATCH 请求所发送的内容都必须是 JSON 格式，并且消息头中必须设置 Content-Type 为 `application/json`，否则 API 会返回一个 `415 Unsupported Media Type` 状态码。

```
$ curl http://10.20.17.211:8008/v1/devices/136 \
-X PATCH \
-H 'Content-Type: application/json' \
-d '{"content":"this is new content"}'
```

HTTP Verbs

我们使用标准的 HTTP 动词来表示一个请求的意图：

- `GET` (选择)：从服务器上获取一个具体的资源或者一个资源列表。
- `POST` （创建）： 在服务器上创建一个新的资源。
- `PUT` （更新）：以整体的方式更新服务器上的一个资源。
- `PATCH` （更新）：只更新服务器上一个资源的一个属性。
- `DELETE` （删除）：删除服务器上的一个资源。

Responses

所有响应的内容都是 JSON 的格式。

单个资源时返回 JSON 对象：

```
{
  "field1": "value",
  "field2": true,
  "field3": []
}
```

资源列表时返回 JSON 对象，数据会被放在了 `items` 对象数组中，`total_count` 是总记录数：

```
{
  "total_count": 135,
  "items": [
    {
      "field1": "value",
      "field2": true,
      "field3": []
    },
    {
      "field1": "another value",
      "field2": false,
      "field3": []
    }
  ]
}
```

如果字段是字符型，使用空字符串（”“）而不是null来表示该字段的值为空。如果该字段是数组，则使用 [] 表示空数组。

HTTP Status Codes

我们使用HTTP状态码来表示一个请求成功或失败。

成功状态码：

- `200 OK` - [GET]：服务器成功返回用户请求的数据
- `201 Created` - [POST/PUT/PATCH]：用户新建或修改数据成功
- `204 No Content` - [DELETE]：用户删除数据成功

错误状态码：

- `400 Bad Request` - [POST/PUT/PATCH]：无法解析用户提交的数据（非JSON格式的数据或JSON数据解析失败），服务器什么也没做
- `404 Not Found` - [*]：用户请求或操作一个不存在的资源或集合，服务器什么也没做
- `415 Unsupported Media Type` - [POST/PUT/PATCH] 当请求头中缺少 `Content-Type: application/json` 时，将返回此错误
- `422 Unprocesable entity` - [POST/PUT/PATCH] 当用户修改或创建一个资源时，发生一个验证错误，服务器什么也没有做
- `500 Internal Server Error` - [*]：服务器发生内部错误，用户将无法判断发出的请求是否成功

Errors

400,404,500 系列的错误，响应内容是一个 JSON 对象，并且响应头的 Content-Type 为 application/json。

- 400 错误

```
{
    "code": "10005",
    "message": "无法解析你所提交的数据"
}
```

- 404 错误，有以下三种可能的返回：

i. 接口不存在时

```
{
    "code": "10020",
    "message": "你所请求的接口不存在"
}
```

ii. 请求的单个资源不存在时

```
{
    "code": "20104",
    "message": "你所请求的设备不存在"
}
```

iii. 请求的资源列表不存在时

```
{
    "code": "10010",
    "message": "没有任何数据"
}
```

- 415 错误

```
{
    "code": "10011",
    "message": "请求类型错误"
}
```

- 500 错误

```
{
    "code": "10001",
    "message": "对不起，服务器发生内部错误，无法得知请求是否成功"
}
```

`code` 是全局唯一的错误代码，可以查看错误代码表，查询所对应的错误的详细含义。

Validation Errors

POST,PUT,PATHCH 请求发生验证错误时，将会返回 `422 Unprocessable Entry` 状态码。响应内容中将是 JSON 格式的错误信息数组。

```
{
  "code" : 10006,
  "message" : "数据验证错误",
  "errors" : [
    {
      "code" : 21102,
      "field" : "meb_name",
      "message" : "预订人姓名不能为空"
    },
    {
       "code" : 21106,
       "field" : "meb_mobile",
       "message" : "预定人手机号的格式不对"
    }
  ]
}
```

Embedding

许多接口的除了返回请求资源自身外，还支持返回与此资源相关的资源，这样可以尽量减少API 调用的次数。

使用一个 `embed` 查询参数来实现这一功能。返回多个相关资源时，请用逗号隔开。

```
GET /inn/100/devices?embed=images,attributes
```

响应内容


```
[
  {
    "device_id": "156",
    "inn_id": "100",
    "cat_id": "8",
    "name": "浪漫满屋主题公寓",
    "images": [
      {
        "img_id": "29",
        "device_id": "156",
        "img_url": "http://img0.bdstatic.com/img/image/shouye/xinshouye/sheji206.jpg"
      }
    ],
    "attributes": [
      {
        "id": "22",
        "device_id": "156",
        "attr_id": "899",
        "attr_value": "有空调，电冰箱"
      }
    ]
        ... other device fields ...
  }
]
```


并非所有接口使用 `embed` 参数都有效，这与资源本身是否存在相关数据也有关。接口文档会对每个接口是否支持 `embed` 参数进行说明。

Pagination

在请求接口返回多条数据时，我们默认将显示前15条数据。当然你可以使用 `per_page` 和 `page` 来控制数据的返回。

```
GET /inn/100/devices?per_page=15&page=2
```

All Pages

为了满足部分接口需要一次性获取所有数据的要求，我们特意增加了 `all_pages` 参数，添加参数 `all_pages=true` 即可返回该接口的所有数据。当然，为避免服务器压力过大，该参数应该尽量少用。

```
GET /inn/100/devices?all_pages=true
```

`all_pages` 的优先级高于 `per_page` 和 `page` 参数，如果同时存在三个参数，只会识别 `all_pages` 而忽略其他的分页参数。

Filtering

对于接口返回的结果，有时候需要进行筛选。你可以使用字段名作为筛选的参数，比如说你希望获

取ID为100这个分店所有已取消的订单，即可使用

```
GET /inn/100/orders?state=canceled
```

Sorting

有些接口对于返回结果需要进行排序，可以使用 `sort` 参数。排序参数采取逗号分隔的形式，每一个字段前都可能有一个负号 `-` 来表示按降序排序。并不是所有字段都可以进行排序，接口文档将会列出每个接口可进行排序的字段。

```
GET /inn/100/devices?sort=-create_time,name
```

Multi Language

对于调用接口返回的错误信息，我们将提供中英两种语言的选择。只需要在请求中设置 `Accept-Language` 报头即可。`Accept-Language: zh-cn` 表示希望返回中文信息，`Accept-Language: en-us` 表示希望返回英文信息，默认返回中文信息。

英文信息返回示例

```
$ curl http://10.20.17.211:8008/v1/device/99999 \
-X PATCH \
-H 'Accept-Language: en-us' \
-d '{"name":"My-new-name"}'
```

响应内容

```
404 Not Found
Content-Type: application/json
Content-Language:en-us

{
    "code": "20104",
    "message": "Target device does not exist"
}
```

Other Features

以下将对在此版本我们可能暂时无法实现（这是基于我们的开发进度和复杂度的考虑，这不代表这些特征是无关紧要的。如果你觉得应该在此版本中提供，可以把你的想法通过邮件告诉我们），但在未来可能实现的接口特征作一些说明。

Field Filtering

有时候 API 使用者不需要返回结果的所有字段，而是部分字段。在进行横向限制的时候（例如只返回API结果的前十条记录）还应该可以进行纵向限制。这个功能能有效减少网络带宽的压力和提高传输速度。实现该功能可以使用 `fields` 参数来筛选返回的字段。

```
GET /inn/100/devices?fields=device_id,name
```

响应内容

```
{
  "total_count": "2",
  "items": [
```

```
    {
      "device_id": "10056",
      "name": "黄河会议室"
    },
    {
      "device_id": "10057",
      "name:": "长江会议室"
    }
  ]
}
```

Enveloping

如果你的 HTTP客户端获取状态码或头信息比较困难，或者你希望使用 JSONP 进行跨域请求时，我们可以把响应的所有内容（包括响应头）全部组装好放在响应内容里。只需要加上 envelope=true 请求参数，而此时接口只会返回 200 状态码。而真正的状态码、头信息以及响应内容都会在组装好的响应内容里（包装在信封里）。

```
GET /inn/100/devices/does-not-exist?envelope=true
```

响应内容

```
200 OK
Content-Type: application/json
Content-Language:zh-cn

{
  "status": 404,
  "headers": {
    "Content-Type": "application/json",
    "Content-Language": "zh-cn"
  },
  "response": {
        "code": "20104",
    "message": "你所请求的设备不存在"
  }
}
```

Changes

新版设备管理系统API 即第二版API 将发生以下变化：

1. 新增获取接口所有数据的分页参数 `all_pages`
2. 错误信息返回支持中英双语 `Accept-Language`
3. 任何的错误返回信息都有一个唯一的错误代码，附错误代码表
4. 取消 `423 Business Error` 错误，所有的验证错误都是 `422 Unprocesable entity`
5. 当使用 POST,PUT,PATCH 操作资源时，必须设置请求头 `Content-Type: application/json`

为了更好的接口体验，我们错误代码的设计，将参考较为完善的微博开发平台 API，点击查看微博API的错误码。

References

1. RESTful 最佳实践
   - http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api （英文原文）
   - http://blog.jobbole.com/41233/ （中文译文1）
   - http://www.oschina.net/translate/best-practices-for-a-pragmatic-restful-api （中文译文2）
2. 好RESTful API的设计原则
   - http://codeplanet.io/principles-good-restful-api-design/ （英文原文）

    - http://www.cnblogs.com/moonz-wu/p/4211626.html （中文译文）
3. Some REST best practices https://bourgeois.me/rest/
4. 理解RESTful架构（阮一峰） http://www.ruanyifeng.com/blog/2011/09/restful.html
5. RESTful API 设计指南（阮一峰） http://www.ruanyifeng.com/blog/2014/05/restful_api.html
6. RESTful Web Services中文版 http://book.douban.com/subject/3094230/
7. RESTful Web Services Cookbook中文版 http://book.douban.com/subject/6837645/
8. RESTful Web APIs中文版 http://book.douban.com/subject/25909247/
9. Github API v3 https://developer.github.com/v3/
10. Mailgun Documentation https://documentation.mailgun.com/
11. Enchant REST API http://dev.enchant.com/api/v1
12. 微博 API http://open.weibo.com/wiki/Error_code
13. 最佳实践：更好的设计你的 REST API
    http://www.ibm.com/developerworks/cn/web/1103_chenyan_restapi/

Whys

在编写此文档时，我觉得直接使用英文更能直接说明我们的目的，所以这里的标题大多直接使用了英文，这对了解REST的开发者也是友好的。

Contact Us

新版本的接口，我们将准备按照本文档进行重新开发。由于改动较大，我们准备将其直接升级到第二版。文档初稿中会有不完善的地方，如果您有任何更好的建议或者一些疑问，希望您能联系我们，谢谢。

您可以把您的想法发送到以下邮箱： 1079087531@qq.com

如果您愿意，您也可以选择 QQ 联系我们：点击弹出qq聊天窗口。