一、restful api

可以总结为一句话：REST是所有Web应用都应该遵守的架构设计指导原则。Representational State Transfer，翻译是"表现层状态转化"。 **面向资源是REST最明显的特征**，对于同一个资源的一组不同的操作。资源是服务器上一个可命名的抽象概念，资源是以名词为核心来组织的，首先关注的是名词。REST要求，必须通过统一的接口来对资源执行各种操作。对于每个资源只能执行一组有限的操作。（7个HTTP方法：GET/POST/PUT/DELETE/PATCH/HEAD/OPTIONS）

1、Restful API设计规范

- **资源**。首先是弄清楚资源的概念。资源就是网络上的一个实体，一段文本，一张图片或者一首歌曲。资源总是要通过一种载体来反应它的内容。文本可以用TXT，也可以用HTML或者XML、图片可以用JPG格式或者PNG格式，JSON是现在最常用的资源表现形式。
- **统一接口**。RESTful风格的数据元操CRUD（create,read,update,delete）分别对应HTTP方法：GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用来更新资源，DELETE用来删除资源，这样就统一了数据操作的接口。
- **URI**。可以用一个URI（统一资源标识符）指向资源，即每个URI都对应一个特定的资源。要获取这个资源访问它的URI就可以，因此URI就成了每一个资源的地址或识别符。一般的，每个资源至少有一个URI与之对应，最典型的URI就是URL。
- **无状态**。所谓无状态即所有的资源都可以URI定位，而且这个定位与其他资源无关，也不会因为其他资源的变化而变化。有状态和无状态的区别，举个例子说明一下，

例如要查询员工工资的步骤为第一步：登录系统。第二步：进入查询工资的页面。第三步：搜索该员工。第四步：点击姓名查看工资。这样的操作流程就是有状态的，查询工资的每一个步骤都依赖于前一个步骤，只要前置操作不成功，后续操作就无法执行。如果输入一个URL就可以得到指定员工的工资，则这种情况就是无状态的，因为获取工资不依赖于其他资源或状态，且这种情况下，员工工资是一个资源，由一个URL与之对应可以通过HTTP中的GET方法得到资源，这就是典型的RESTful风格。

- **RESTful API还有其他一些规范。**
  - 应该将API的版本号放入URL。
    GET:http://www.xxx.com/v1/friend/123。或者将版本号放在HTTP头信息中。版本号取决于自己开发团队的习惯和业务的需要，不是强制的。
  - URL中只能有名词而不能有动词，操作的表达是使用HTTP的动词
    GET,POST,PUT,DELETEL。URL只标识资源的地址，既然是资源那就是名词了。
  - 如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。?limit=10：指定返回记录的数量、?page=2&per_page=100：指定第几页，以及每页的记录数。

2、到底什么是RESTful架构

- 每一个URI代表一种资源
- 客户端和服务器之间，传递这种资源的某种表现层
- 客户端通过四个HTTP动词，对服务端资源进行操作，实现"表现层状态转换"

3、HTTP常用动词

- GET（SELECT）：从服务器取出资源
- POST（CREATE or UPDATE）：在服务器创建资源或更新资源
- PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）
- PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）
- DELETE（DELETE）：从服务器删除资源
- HEAD：获取资源的元数据
- OPTIONS：获取信息，关于资源的哪些属性是客户端可以改变的

示例

- GET /students：获取所有学生
- POST /students：新建学生
- GET /students/id：获取某一个学生
- PUT /students/id ：更新某个学生的信息（需要提供学生的全部信息）
- PATCH /students/id：更新某个学生的信息（需要提供学生变更部分信息）
- DELETE /students/id：删除某个学生

4、restful相关的网络请求状态码

- 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
- 202 Accepted - [*]：表示一个请求已经进入后台排队（异步任务）
- 204 NO CONTENT - [DELETE]：表示数据删除成功
- 400 INVALID REQUEST - [POST/PUT/PATCH]：用户发出的请求有错误
- 401 Unauthorized - [*]：表示用户没有权限（令牌，用户名，密码错误）
- 403 Forbidden - [*]：表示用户得到授权，但是访问是被禁止的
- 404 NOT FOUND - [*]：用户发出的请求针对的是不存在的记录
- 406 Not Acceptable - [*]：用户请求格式不可得
- 410 Gone - [GET]：用户请求的资源被永久移除，且不会再得到的
- 422 Unprocesable entity -[POST/PUT/PATCH]：当创建一个对象时，发生一个验证错误
- 500 INTERNAL SERVER EROR - [*]：服务器内部发生错误

二、重量级RESTful框架

1、web应用模式：

- **前后端不分离**

  在前后端不分离的引用模式中，前端页面看到的效果都是由后端控制的，由后端页面渲染或者重定向，也就是后端需要控制前端的展示，前端与后端的耦合度很高，这种模式比较适合纯网页应用，但是后端对接APP时，App可能并不需要后端返回一个HTML网页,仅仅是数据本身,所以后端原本返回网页的接口不再适用前端APP应用,为了对接APP后端还需再开发一套接口。

- **前后端分离**

  在前后端分离的应用模式中，后端仅返回前端所需要的数据，不再渲染HTML页面，不再控制前端的效果，只要前端用户看到什么效果，从后端请求的数据如何加载到前端中，都由前端自己决定，网页有网页自己的处理方式，APP有APP的处理方式，但无论哪种前端所需要的数据基本相同，后端仅需开发一套逻辑对外提供数据即可，在前后端分离的应用模式中,前端与后端的耦合度相对较低

2、Django Rest Framework

Django Rest Framework （DRF）是一个强大且灵活的工具包，用以构建Web API。 Django REST Framework可以在Django的基础上迅速实现API，并且自身还带有WEB的测试页面，可以方便的测试自己的API。

- 特性
    - 可浏览API
    - 提供丰富认证
    - 支持数据序列化
    - 可以轻量嵌入，仅使用fbv
    - 强大的社区支持

官方网站：

```
https://www.django-rest-framework.org/
```

中文翻译网站:

```
https://q1mi.github.io/Django-REST-framework-documentation/
```

安装djangorestframework

```
pip install djangorestframework
```

3、django-rest-framework内容

- 序列化
- request和response对象
- 基于类的视图
- 认证和鉴权