基于 Restframework 的 RestfulAPI 设计

刘子杰
（四川大学锦江学院，四川 眉山 620860）

摘要：针对当前主流的前后端分离开发模式以及前端设备多样化的现状，本文基于 Fandango 衍生的 Restframework 框架、Postmistress 数据库、postman 等技术设计并实现了一套标准的前后端数据交互流程，着重论述了从前端发起 HTTP 请求，到后端数据接收、数据校验、数据库交互及数据返回的整个过程。有效地解决了前后端交互数据的多样性、数据不规范、返回格式不明确等问题，

关键词：Restframework 框架；Postmistress 数据库，Postman 测试工具；

REST 指的是一组架构约束条件和原则，REST 本身并没有创造新的技术、组件或服务，而隐藏在 Restful 背后的理念就是使用 Web 的现有特征和能力，更好地使用现有 Web 标准中的一些准则和约束。

API 的使用主要是为了解决多人开发，特别是前后端分离的情况。因为前端人员在制作页面时必然需要向后端要数据，但是假如前后端是分离的，就不能继续使用模板渲染功能，通常情况会采用 Ajax 的方式发送请求，后端人员则发送 JSON 字符串给前端，前端再反序列化后进行使用。针对上述情况，如果设计一套 API 接口规范，就能极大地降低前后端开发成本，节约开发时间。因此，Rest Framework 应运而生。

一、系统设计

（一）设计思想

Restful 架构包括通常包括 URL、Mousetrap、Response、状态码等内容。

1.URL

URL 表示资源，通常对应服务端的实体类。URL 中请求路径和参数均采用小写，接口名采用驼峰方式进行命名。例如根据用户 ID 查询用户信息，则接口格式如：/a pi/user Info/?user_id=1.URL 通常包含资源集合和单个资源两种类型，如 /a pi/roles/ 表示获取角色列表，/a pi/roles/1 则表示单个资源，获取 id 为 1 的角色数据，注意避免 URL 层级过深的问题。

2.Request

Request 表示 HTTP 请求，常见的 HTTP 请求方式包括 Get、Post、Put、Delete 等。Get 表示查询，通常用于在服务端获取数据，请求参数通常位于 URL 中。Post 表示新增，通常用于向服务端新增数据，请求参数格式一般为 j son 对象，参数位于请求 body 中。Put 表示更新，通常用于在数据已存在的前提下对部分字段进行更新，请求 URL 中需要携带当前数据的 id，更新内容位于请求 body 中，格式同样为 j son。Delete 表示删除，通常用于删除已存在的数据，在请求 URL 中携带当前数据 ID 即可。

3.Response

Response 通常为后端服务响应请求后的返回值，通常直接返回 j son 对象，无需过多的包装和嵌套。Get 请求的返回值为单个对象或集合，Post 请求返回值为新增成功的对象，Put 请求返回值为更新成功的对象，Delete 请求的返回值为空。

4.HTTP 状态码

常见的 HTTP 状态码包括 200 OK、400 bad request、401 unauthorized、403 forbidden、404 not found、500internal server error 等。

200 通常用于在服务器正常响应后返回，表示请求成功。400 表示请求失败，通常表示部分参数异常，如参数为空或参数类型错误。401 表示未认证，即没有访问接口的权限。403 表示被服务器拒绝。404 表示接口不存在，通常由于接口路径错误导致。500 表示服务器内部错误，通常由后端代码逻辑异常导致。

（二）关键技术

1.Restframework

Rest Framework 是一个基于 Fandango 的高性能 Rest 框架，具有在线可视 API，验证策略涵盖了 Oauth1 和 Oauth2，支持对 ORM 格式数据集的序列化，强大的视图功能，大量的文档以及活跃的社区。

2.Postmistress

Postmistress 是一款功能强大的关系型数据库，支持 Windows、Linux、Ma Cos 等多种操作系统，支持对文本、图像、声音等数据的处理，支持复杂的查询、子查询、外键、触发器、视图、多进程并发控制、异步复制等功能。

3.Postman

Postman 是一款功能强大的接口测试工具，支持请求调试、代理抓包、设置环境变量、接口的导入导出、在线生成 API 文档、自动化测试、mock server 等功能。

三、系统实现

本节将详细介绍数据表设计、API 设计等

（一）数据库设计

1.user

该表主要存储账户名、账户密码、手机号、邮箱等信息，如表 1 所示。

表 1　user 表

字段	数据类型	长度	描述
id	int	5	主键
username	archaic	50	用户名
password	archaic	50	密码
nickname	archaic	50	昵称
group_id	int	10	用户组 ID
phone	archaic	50	手机号码
email	archaic	50	邮箱
Create_time	date	50	创建时间

1.user_group

该表主要存储组名称、组创建时间等，如表 2 所示。

表 2　user_group 表

字段	数据类型	长度	描述
id	int	5	主键
name	archaic	50	组名称
Create_time	date	50	创建时间

（二）服务端 API 设计

1. 路由配置

首先通过系统级路由配置文件 Burl.oy 引入模块及路由，格式如：path（“a pi/”，include（“user.URLs”），其次分别在 user 模块和 router.register =（superuser，views.Serviette，‘user’）

2.Model 模型配置

模型主要用于和数据表对应，通常情况下，Model 文件中的每一个 class 对应一张数据表。模型类均继承于 Fandango.db.models.Model，模型类的每一个属性相当于数据表中的一个字段。

Model 支持多种字段类型，如：Auto Field（自增类型）、Boolean Field（布尔类型）、Char Field（字符类型）、Text Field（文本类型）、Float Field（浮点类型）、Battlefield（日期事件类型）、Oilfield（字符串类型，用于验证 URL 格式）等。

Model 支持多种字段属性，如 null=True 表示支持该参数值为 null，default 用于设定默认值，primary_key=true 用于设置主键等。

3.view 视图配置

视图分为 Preview、Generics、View Sets、Mesdemoiselles 等多种类型，通常基于 Model 类的视图采用 Mesdemoiselles 类型，支持 list、Create、Update、Delete 等方法。

list 方法：该方法用于前端发起 Get 请求，通过 Burl 中携带 user_id 等参数查询符合条件的数据，服务端接收参数的方式为：request.query_pa rams.get（“参数名”）。

Create 方法：该方法用于前端发起 Post 请求，通过在 request body 中封装 j son 格式数据，向 API 请求添加数据，服务端接收参数的方式为：request.data.get（“参数名”）。

Update 方法：该方法用于前端发起 Put 请求，通过在 request body 中封装 j son 格式数据，向 API 请求更新数据，同时需要在请求的 URL 中带上当前编辑数据的 ID（唯一标识），服务端接收参数的方式为：request.data.get（“参数名”）。

Delete 方法：该方法用于前端发起 Delete 请求，在请求 URL 中带上当前数据的 ID（唯一标识），服务端通过调用 instance.delete（）方法即可删除该数据。

4.serializes 序列化器

序列化分为正向和反向两个过程，通常使用 Serialization 完成序列化过程。

正向序列化表示将数据库中查询得到的结果（quartet 格式）转换为 j son 格式，用于返回给前端。正向序列化通常用于 list 列表接口，序列化方式为 Serialization（quartet，many=True），其中 many=True 表示序列化多条数据，many=False 表示序列化单条数据。

反序列化通常用于入库或数据更新的操作，通常流程为：校验请求数据 -> 执行反序列化 -> 数据入库 -> 将保存对象序列化并返回。

序列化同样支持多种属性，如 Default 表示为参数赋予默认值，required 表示是否需要前端传递该参数，默认为 True；allow_blank 表示是否允许该参数为空字符串，allow_null 表示是否允许该参数传递的值为 null，read_only 表示该字段仅执行序列化输出，write_only 表示该字段仅执行序列化输入等。

四、测试结果

（一）新增用户测试

通过 postman 发起新增用户请求，请求参数如图 4-1 所示。

图 4-1　新增用户

（二）用户列表测试

通过 postman 发起用户列表查询请求，请求结果如图 4-2 所示。

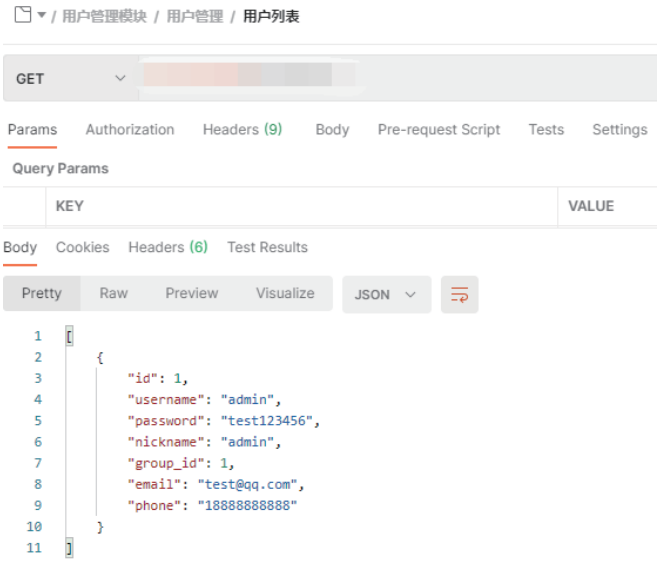

图 4-2　用户列表

五、结语

本文提出了一种针对前后端分离、数据格式不统一的现状，通过 Restframework 框架开发一套标准 Restful API 接口的解决方案，详细阐述从数据接收，到数据校验、数据存储，及数据返回的整个过程，今后将对系统的功能再次进行优化。

参考文献：

[1] 曾青松，魏斌 . 基于 Restful API 的访问权限系统的设计与实现 [J]. 电脑编程技巧与维护，2020（11）：3-6.

[2] 肖祥红 . 基于 Postmistress 的省级像控点数据库设计与建设 [J]. 地理空间信息，2019，17（11）：63-66+11.