REST API design best practices â€“ Part 2

Description

Introduction

In this post we are going to see some REST API design best practices. Most of the developers underestimate the design aspect while working with REST APIs. There are a lot of ways in which we can design REST APIs, but we are going to cover all of them in a series of blog post just to keep each blog post short and concise.

If you haven't gone over the previous blog post on basic REST API design best practices then I would like you to have a quick glance. In this post we are mostly going to cover various scenarios of passing Request parameters for additional options or metadata for operations such as:

- filter
- sort
- pagination

Rule#1 – Filtering data using key-value

If you want to filter data using a single key value pair or multiple key value pairs then you can use Request param as shown below in which we are fetching the records whose **lastname** is **Doe** and **age** is **51**.

/customers?lastname=Doe&age=51 

Rule#2 – Filtering data for specific fields

In **Rule#1** we saw how to fetch the records for a specific criteria, which returns all the keys. But what if we want to fetch records which contain only specific fields? in that case we can use below approach. Here we are fetching all the customer records which contain only data for **firstname, lastname and age**.

/customers?fields=firstname,lastname,age 

This is similar to fetching specific columns from the database.

Rule#3 – Limiting the data returned

If you want to limit the number of items for the data returned then we can use below approach. In the below example we are fetching only 10 order records by passing the key as **limit** and value as **10**.

/customers/{customer_id}/orders?limit=10 

Rule#4 – Pagination

In order to paginate the data chunk by chunk instead of querying all at once we can use below rule. Below example will return the 50 records starting from 0. if you want to return the next chunk we make the start as 51 and limit as 50 or whatever number of records you want.

/customers?start=0&limit=50 

Rule#5 – Sorting

If you want to sort the data by specific key-value pair then you can use request param as shown below. This example sorts the returned data based on price.

/customers/{customer_id}/orders?sort=price 

Rule#6 – Logical grouping of collection

We should leverage logical grouping by reflecting object relationship. Example can be customers have orders.

…/customers/{customer_id}/orders/{order_id} 

Avoid reflecting database structure in designing APIs.

Conclusion

I hope you liked the part two of our REST API design best practices series. This post uncovered some unique rules when it comes to filtering, sorting and paginating the records. Please share this post as much as possible if you found it insightful.
Happy Learning ðŸ˜Š

Category

1. Design

Date Created
January 2, 2024
Author

kk-ravi144gmail-com