Cheatography

Common Practices for REST API Development Cheat Sheet

by Paladuta Stefan via cheatography.com/139860/cs/30803/

Accept and respond with JSON

JSON is the standard for transferring data. Almost every networked technology can use it: JavaScript has built-in methods to encode and decode JSON either through the Fetch API or another HTTP client. Server-side technologies have libraries that can decode JSON without doing much work.

To make sure that when our REST API app responds with JSON that clients interpret it as such, we should set `Conten t-Type` in the response header to `applic ati on/json` after the request is made. Many server-side app frameworks set the response header automatically. Some HTTP clients look at the Content-Type response header and parse the data according to that format.

NOTE: Spring Boot offers great support for responding with json or xml format, without effort.

Use nouns instead of verbs in the endpoint path

Nouns represent the entity that we are trying to recover or manipulate in pathname. Verbs should be avoided because they are already present, they are the HTTP requests types: `GET`, `POST`,`DEL ETE`, `PUT`, etc.

Adding verbs in API pathname doesn't add more value, it makes the pathname even more longer and doesn't help in the description of the API.

Good examples:

```
[GET] /api/books
[GET] /api/book
[POST] /api/book
```

Bad examples:

```
[GET] /api/g etA llBooks
[GET] /api/g etBook
```

Using plural in resources naming

We should name collections with the plural form of the nouns, because it's very common when you want to recover a list of resources to also want to recover one resource from that list and vice versa.

We use the plural form to be consistent with what is in your database. A table has more than one entity/record and we as developers should name our endpoint in such a way to reflect this, or more formally to be consistent.

Example(s):

`/api/b ook /{id}` and `/api/books`

`/api/a cco unt /{id}` and `/api/a ccounts`

`/api/p ers on/{id}` and `/api/p ersons`

Versioning the API

Allow filtering, sorting & pagination

A database behind a REST API can be very big and that means that it's not always a good idea to return everything to the client, because it may slow down the system. In these cases we must offer the client some filtering options based on what he specifically wants.
Filtering and also pagination grows the performance of your REST API because it doesn't make the system to compile the entire database of resources in the response.

Example:

```
http:/ /ex amp le.c om /ar tic les /?s ort =+a uth
 or, -da tep ubl ished
```

Where `+` means ascending.

Where `-` means descending.

A response body should never contain HTTP info

We should avoid completely putting HTTP information into the response body.

We don't need to tell the caller of the API in the response body that the API worked with 200 OK or 404 NOT FOUND (etc.).

Our API should be capable in case of error to fail with an HTTP correct status. And not offer confusing information in the response body like failing with 200 OK

Example of bad response

```
{
      " sta tus " : " 200 ",
      " res ponse: " OK",
      " id" : 1,
      " nam e" : Stefan
      " age " : 25
}
```

Use and maintain good security practices

Use SSL/TLS for security.
Introducing a SSL certificate on a server is not difficult, actually presents a really low cost and there is no reason not to make our API that can be called publicly (even private) secured.

We should always version our API's as good practice. Versioning is something that clients appreciate and offers them more trust when using your API, because you will not enforce modification on their side for each major implementation update on your API, but instead they are going to migrate to your new version only when they are ready.

Example:
`https: //a ppg ate way.li dlp lus.co m/a pp/ v19 /R O /ti cke ts/ list/` where v19 shows the version number.

By **Paladuta Stefan**
cheatography.com/paladuta-stefan/

Published 2nd November, 2022.
Last updated 2nd November, 2022.