**REST API Design: Best Practices**

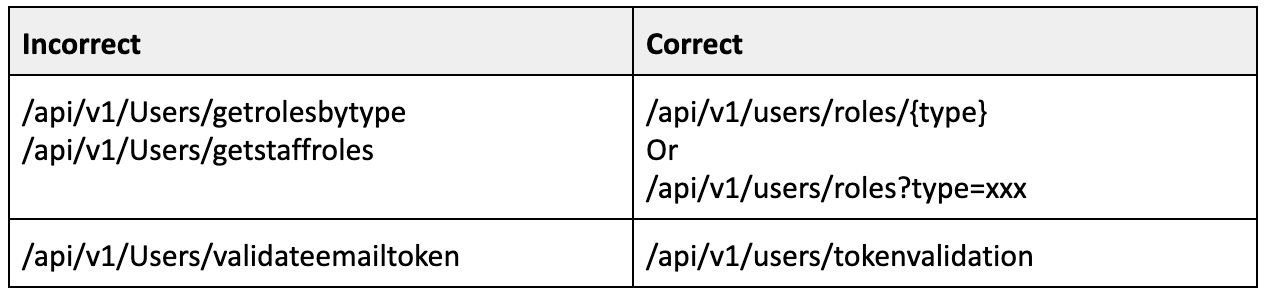
In general, all the developers and the architects follow their own way of designing REST APIs. In this note, I have tried to explain some industry-standard approaches and best practices to design and develop REST APIs and maintain consistency in writing code.

## REST Resource Naming Best Practices

In case of REST APIs, primary data representation is called Resource. Having a strong and consistent REST resource naming strategy — will definitely prove one of the best design decisions in the long term. A resource can be a singleton or a collection. For example, “users” is a collection resource and “user” is a singleton resource. We can identify “users” collection resources using the URI “/users”. We can identify a single “user” resource using the URI “/users/{id}”.

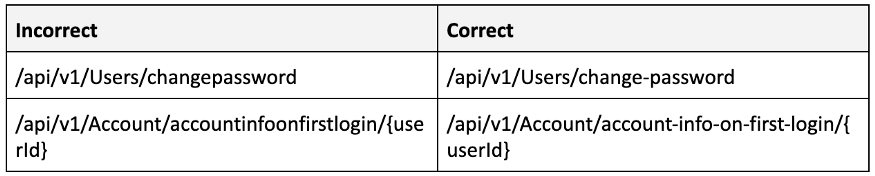
## Use nouns to represent resources

RESTful URI should refer to a resource that is a thing (noun) instead of referring to an action (verb) because nouns have properties that verbs do not have — similar to resources have attributes. Some examples of a resource are:



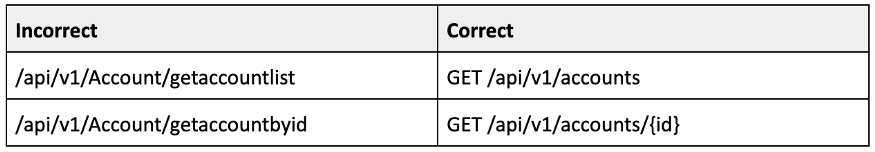
## Use hyphens (-) to improve the readability of URIs

To make your URIs easy for people to scan and interpret, use the hyphen (-) character to improve the readability of names in long path segments.



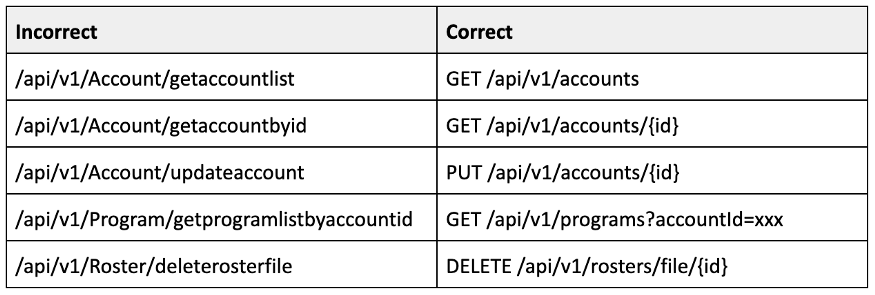
## Use lowercase letters in URIs

When convenient, lowercase letters should be consistently preferred in URI paths.



## Never use CRUD function names in URIs

URIs should not be used to indicate that a CRUD function is performed. URIs should be used to uniquely identify resources and not any action upon them. HTTP request methods should be used to indicate which CRUD function is performed.



## Use query component to filter URI collection

Many times, you will come across requirements where you will need a collection of resources sorted, filtered, or limited based on some certain resource attribute. For this, do not create new APIs — rather enable sorting, filtering, and pagination capabilities in resource collection API and **pass the input parameters as query parameters**. e.g.

