REST APIs

Representational State Transfer, popularized by Roy Fielding in the 2000s, came out as a viable alternative to traditional ways of developing web services in service-oriented architecture (SOA). REST rose to popularity because of its ease of use and its tendency to defer implementation decisions to the developer, in other words REST is an unopinionated style of architecture based in http protocol.

One of the fundamental rules of application development is loose coupling. In a modular architecture, components and services can be swapped out naturally if coupling is kept minimal. REST has proven to be the de facto industry standard to define the ground for client/server and interservice communications. Multiple clients can talk to the same server by relying on exchanging information over a network in any acceptable data format, XML and JSON being the most widely used. REST is stateless, meaning the context of the client is not stored in the server and subsequent requests need to provide their context.

A RESTful web service may expose different types of operations based on the actions a client needs to perform on the resource being served. Four of the major operations are discussed below

GET – A get request is one of the simplest operations. In layman’s terms, a client commonly a browser makes a request to a server to be served up a content. The server checks if the client is a known user with valid permissions and if the resource is available, then sends a response. The server might respond with defined categories of error codes in case of having the above preconditions unfulfilled. The error code response might suggest an unauthorized user, or a resource not found to say the least. More on error codes later. A get request is supposed to be designed to not have side effects of modifying a resource.

POST – A post operation is used for creating a new resource. The data to be added is commonly obtained using an html form or through another client service but getting data through other testing tools like CURL, POSTMAN and SOAPUI are also common. The API may do some validation based on the implementation such as checking for a duplicate resource or if all the required fields are populated and with the correct format before storing the resource.

PUT – A put request is made to modify an existing resource. The implementation might pull the resource based on a parameter passed in from the client and make the appropriate modifications. Unlike a post request, put requests are idempotent, meaning trying to modify a resource that has already been modified doesn’t change anything.

DELETE – A delete request is made when trying to delete a resource. If persistence is part of the application, the resource will be deleted from disk. Trying to delete a resource that doesn’t exist, or other constraints may throw exceptions based on the implementation.

The following categories of error codes can provide a hint to what the server is having trouble with.

1XX – informational. Error code of the 100s provides the client additional information at times like when a request is being processed and can be safely ignored after the process is complete.

2XX – success. The classic 200 error code is used when a request is successful. 204 is also commonly used when a delete request is the successful and there is no content to displayed back in response.

3XX – redirection. Further action needed from user to complete request.

4XX – client error. The most common 404 error code is displayed when client tries to request a request that doesn’t exist. 403 is also common when a user is not authorized to obtain a resource.

5XX – server error. The most common 500 error code is displayed when the server is not available to serve resource. This is a generic response which may result from different types of outages.

Express.Js is one of the many JavaScript frameworks and libraries available that speed up development of RESTful web services. Most of the standard server-side languages also have addons to reduce the boilerplate code required to setup a RESTful web service.

More advanced topics like Hypermedia as the Engine of Application, HATEOAS, will be discussed in later iterations.

# Bibliography

<http://www.looah.com/source/view/2284>

<https://itnext.io/javascript-fundamentals-an-introduction-to-rest-apis-7cbe8a809d3b>

<https://restfulapi.net/idempotent-rest-apis/>

<https://www.restapitutorial.com/httpstatuscodes.html>