REST API

**What is REST API?**

**What are the constraints in REST API?**

REST defines 6 architectural constraints which make any web service – a true RESTful API.

* Uniform interface
* Client–server
* Stateless
* Cacheable
* Layered system
* Code on demand (optional)

**What's the difference between REST & RESTful?**

<https://stackoverflow.com/questions/1568834/whats-the-difference-between-rest-restful>

<https://devmountain.com/blog/what-is-the-difference-between-rest-and-restful-apis/#:~:text=Put%20simply%2C%20there%20are%20no,services%2C%20applications%2C%20and%20software>.

REST stands for representational state transfer. It is a set of constraints that set out how an API (application programming interface) should work. If an API is RESTful, that simply means that the API adheres to the REST architecture. Put simply, there are no differences between REST and RESTful as far as APIs are concerned. REST is the set of constraints. RESTful refers to an API adhering to those constraints. It can be used in web services, applications, and software.

**What is a resource?**

A resource is how data is represented in the REST architecture. By exposing entities as the resource, it allows a client to read, write, modify, and create resources using HTTP methods, for example, [GET](http://javarevisited.blogspot.sg/2012/03/get-post-method-in-http-and-https.html), [POST](http://www.java67.com/2014/08/difference-between-post-and-get-request.html), [PUT](http://www.java67.com/2016/09/when-to-use-put-or-post-in-restful-web-services.html), DELETE, etc.

**Is REST normally stateless?**

Yes, REST API should be stateless, because it is based on HTTP, which is also stateless. A request in REST API should contain all the details required to process it.

It should not rely on previous or next requests or some data maintained at the server end, like sessions.

The REST specification puts a constraint to make it **stateless**, and you should keep that in mind while designing your REST API.

**Are There Other API Styles Besides REST?**

Prior to 2000, APIs were the Wild West, and there were many different types. Since then, REST has risen in popularity, becoming somewhat of the go-to API style architecture. That being said, there are other API styles. [SOAP](https://en.wikipedia.org/wiki/SOAP) (Simple Object Access Protocol) is another API style. Depending on what your project goals are, one API style might be better than another for you.

**What Constraints Are Set Out By REST?**

There are 4 main principles of REST as laid out by Roy Fielding and his colleagues in 2000. They set out to create a standard that allowed servers to communicate with other servers easily. This is what they came up with, changing the landscape of APIs:

1. Client-Server: There is always a client and a server, and these two systems need boundaries for how they operate. Which one is being called (server) and which one is making the request (client)? Having these boundaries leads to smoother operation.
2. Stateless: Servers need to be able to process messages they receive. In order to do this, every request a server receives should have the necessary information required for the server to work.
3. Uniform Interface: Using similar terminology and resources helps standardize APIs. According to this principle, the following HTTP verbs are used: GET, PUT, POST, and DELETE. Resources always refer to URIs (uniform resource identifier). HTTP responses always come with a status and a body.
4. Cacheable: Clients need to be able to cache representations. Because of statelessness (every representation being self-descriptive), this is possible in a RESTful API.

**If REST applications are supposed to be stateless, how do you manage sessions?**

<https://stackoverflow.com/questions/3105296/if-rest-applications-are-supposed-to-be-stateless-how-do-you-manage-sessions?rq=1>

# How to manage state in REST?

<https://stackoverflow.com/questions/2641901/how-to-manage-state-in-rest?noredirect=1&lq=1>

# What is idempotent in REST API?

* <https://www.restapitutorial.com/lessons/idempotency.html#:~:text=From%20a%20RESTful%20service%20standpoint,as%20making%20a%20single%20request>.
* <https://blog.dreamfactory.com/what-is-idempotency/>
* https://restfulapi.net/idempotent-rest-apis/

## What are idempotent operations? Why is idempotency important?

There are some HTTP methods — like GET — that produce the same response no matter how many times you use them, sending multiple GET request to the same [URI](http://www.java67.com/2013/01/difference-between-url-uri-and-urn.html) will result in the same response without any side-effect. Hence, this is known as idempotent.

On the other hand, the [POST is not idempotent](http://javarevisited.blogspot.sg/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html), because if you send multiple POST requests, it will result in multiple resource creation on the server, but, again, PUT is idempotent, if you are using it to update the resource.

Even multiple PUT requests can be used to update a resource on a server and will give the same end result. You can take an [**HTTP Fundamentals**](http://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fxhttp-fund) course by Pluralsight to learn more about idempotent methods of the HTTP protocol and HTTP in general.

**What is the difference between POST and PUT in HTTP?**

<https://stackoverflow.com/questions/630453/what-is-the-difference-between-post-and-put-in-http?rq=1>

**What is the use of PUT vs PATCH methods in REST API real life scenarios?**

<https://stackoverflow.com/questions/28459418/use-of-put-vs-patch-methods-in-rest-api-real-life-scenarios?rq=1>

**Should I use PATCH or PUT in my REST API?**

https://stackoverflow.com/questions/24241893/should-i-use-patch-or-put-in-my-rest-api?rq=1

**What is the difference between PUT and PATCH?**

**Understanding REST: Verbs, error codes, and authentication?**

<https://stackoverflow.com/questions/2001773/understanding-rest-verbs-error-codes-and-authentication?rq=1>

**What are the HTTP Status codes?**

<https://restfulapi.net/http-status-codes/>

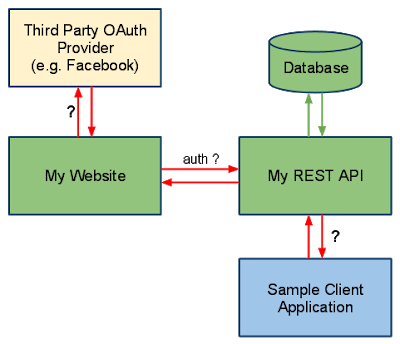
HTTP defines these standard status codes that can be used to convey the results of a client’s request. The status codes are divided into five categories.

* [**1xx: Informational**](https://restfulapi.net/http-status-codes/#1xx) – Communicates transfer protocol-level information.
* [**2xx: Success**](https://restfulapi.net/http-status-codes/#2xx) – Indicates that the client’s request was accepted successfully.
* [**3xx: Redirection**](https://restfulapi.net/http-status-codes/#3xx) – Indicates that the client must take some additional action in order to complete their request.
* [**4xx: Client Error**](https://restfulapi.net/http-status-codes/#4xx) – This category of error status codes points the finger at clients.
* [**5xx: Server Error**](https://restfulapi.net/http-status-codes/#5xx) – The server takes responsibility for these error status codes.

## 10. Is REST secure? What can you do to secure it?

This question is mostly asked by experienced Java programmers with about 2 to 5 years of experience with both REST and Spring. Security is a broad term; it could mean the security of a message, which is provided by encryption or access restriction that is provided using authentication and authorization.

REST is normally not secure, but you can secure it by using Spring Security. At the very least, you can enable the HTTP basic authentication by using HTTP in your Spring Security configuration file. Similarly, you can expose your REST API using[HTTPS](http://javarevisited.blogspot.sg/2013/07/how-ssl-https-and-certificates-works-in-java-web-application.html), if the underlying server supports HTTPS.

[[](https://medium.com/javarevisited/why-spring-is-the-best-framework-for-developing-rest-apis-in-java-784590e484a4?source=collection_home---4------0-----------------------)](https://medium.com/javarevisited/why-spring-is-the-best-framework-for-developing-rest-apis-in-java-784590e484a4?source=collection_home---4------0-----------------------)

## 11. Does REST work with transport layer security (TLS)?

Transport Layer Security (TLS) is used for secure communication between the client and the server. It is the successor of SSL (Secure Socket Layer). Since HTTPS can work with both SSL and TLS, REST can also work with TLS.

Actually, in REST, it is up to the server to implement security protocols. The same RESTful web service can be accessed using HTTP and HTTPS if the server supports [SSL](http://javarevisited.blogspot.sg/2013/07/how-to-configure-https-ssl-in-tomcat-6-7-web-server-java.html#axzz56WXxxAC0).

If you are using Tomcat, you can learn more about how to enable SSL in Tomcat.

**What is REST API versioning? When and how to version REST API?**

<https://restfulapi.net/versioning/>

# rest vs soap (soap structure is like envlope,body, rest is having few infos)

# get call using post

# using post get called

# We can do get into post( if u r searching something in google, acutually it’s updating into db)