What is a RESTful API? A Complete Guide

No comments

REST API and RESTful API are both terms that can be used interchangeably. REST stands for Representational State Transfer and it is a software architectural style that defines a set of certain rules or constraints for creating web services. API is the acronym for Application Programming Interface. REST API is a kind of API that uses the rules defined by the REST architectural style and is also a way of accessing web services in a simple and flexible way without having any processing. Web services that implement REST architecture are called RESTful web services.

What is an API?

An API or an Application Programming Interface is a kind of a gateway between the clients and the resources on the web. These resources contain certain data that can be accessed by rules defined in the API. In other words, API is a way for various applications to communicate with each other. Developers expose or create APIs so that other applications can communicate with their applications programmatically. For instance, suppose you want to create a software that any user can enter their bank account balance and see it in the Bitcoin unit (see how much they will have in Bitcoin if they change all their balance to bitcoin), you will then need an API for retrieving the current price of the BTC at any time. Coin Market Cap API is one of these APIs that can give you the price of BTC at any time. Using this API you can programmatically convert the user's balance to BTC so that he or she can see it in the BTC unit.

What is a RESTful API?

RESTful API is a kind of application programming interface that two computer systems use to exchange data securely over the internet. Most business applications have to communicate with other internal and third-party applications to perform various tasks. For instance, to generate monthly payslips, your internal accounts system has to share data with your customer's banking system to automate invoicing and communicate with an internal timesheet application. RESTful APIs support this information exchange because they follow secure, reliable, and efficient software communication standards. These communication standards or the constraints that we previously mentioned need to be applied on the API to be considered as a RESTful API, otherwise this API will lack the standards of a typical RESTful API.

REST Architecture Standards (Constraints)

There are several principles that an API must have in order to be considered a RESTful API. These standards include:

1. Uniform interface
2. Statelessness
3. Layered system
4. Cacheability
5. Code on demand (optional)

All of the above constraints are the standards of the REST architectural style. The 5th one is optional, meaning that if the API lacks it, it will still be considered RESTful. We will take a look at all of these standards one by one to what each of them actually means:

Uniform interface

The uniform interface is fundamental to the design of any RESTful web service. It indicates that the server transfers information in a standard format. The formatted resource is called a representation in REST. This format can be different from the internal representation of the resource on the server application. For example, the server can store data as text but send it in an HTML representation format.

1. Uniform interface imposes four architectural constraints:

2. Requests should identify resources. They do so by using a uniform resource identifier.

3. Clients have enough information in the resource representation to modify or delete the resource if they want to. The server meets this condition by sending metadata that describes the resource further.

4. Clients receive information about how to process the representation further. The server achieves this by sending self-descriptive messages that contain metadata about how the client can best use them.

5. Clients receive information about all other related resources they need to complete a task. The server achieves this by sending hyperlinks in the representation so that clients can dynamically discover more resources.

Statelessness

In REST architecture, statelessness refers to a communication method in which the server completes every client request independently of all previous requests. Clients can request resources in any order, and every request is stateless or isolated from other requests. This REST API design constraint implies that the server can completely understand and fulfill the request every time.

Layered system

In a layered system architecture, the client can connect to other authorized intermediaries between the client and server, and it will still receive responses from the server. Servers can also pass on requests to other servers. You can design your RESTful web service to run on several servers with multiple layers such as security, application, and business logic, working together to fulfill client requests. These layers remain invisible to the client.

Cacheability

RESTful web services support caching, which is the process of storing some responses on the client or on an intermediary to improve server response time. For example, suppose that you visit a website that has common header and footer images on every page. Every time you visit a new website page, the server must resend the same images. To avoid this, the client caches or stores these images after the first response and then uses the images directly from the cache. RESTful web services control caching by using API responses that define themselves as cacheable or noncacheable.

Code on demand (optional)

In REST architectural style, servers can temporarily extend or customize client functionality by transferring software programming code to the client. For example, when you fill a registration form on any website, your browser immediately highlights any mistakes you make, such as incorrect phone numbers. It can do this because of the code sent by the server.

How do RESTful APIs work?

The basic function of a RESTful API is the same as browsing the internet. The client contacts the server by using the API when it requires a resource. API developers explain how the client should use the REST API in the server application API documentation. These are the general steps for any REST API call:

1. The client sends a request to the server. The client follows the API documentation to format the request in a way that

the server understands.
2. The server authenticates the client and confirms that the client has the right to make that request.
3. The server receives the request and processes it internally.
4. The server returns a response to the client. The response contains information that tells the client whether the request was successful. The response also includes any information that the client requested.

The REST API request and response details vary slightly depending on how the API developers design the API.

Conclusion

In this article, you learned about the APIs, REST architectural style and its principles, and finally the RESTful API. Of course, there is a lot to discuss about these APIs such as the requests and the responses in a RESTful API and the authentication methods. We will talk about these topics in the next articles. Moreover, there are frameworks like Django and Flask using which you can create RESTful APIs like the CRUD API (Create, Read, Update, Delete). We have covered the codes and the implementations of these APIs in Python on the blog tutorials section where you can head over and visit our tutorial contents.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.

Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP NEWSLETTER RSS FEED