**What is REST API?**

REST or **REST**ful stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style as well as an approach for communications purposes that is often used in various web services development. In simpler terms, it is an application program interface (API) that makes use of the HTTP requests to GET, PUT, POST and DELETE the data over WWW.

REST architectural style helps in leveraging the lesser use of bandwidth which makes an application more suitable for the internet. It is often regarded as the “***language of the internet***”. It is completely based on the resources where each and every component is regarded as a component and a single resource is accessible through a common interface using the standard HTTP method.

To understand better, let’s dive a little deeper and see how exactly does a REST API work. Basically, the REST API breaks down a transaction in order to create small modules. Now, each of these modules is used to address a specific part of the transaction. This approach provides more flexibility but requires a lot of effort to be built from the very scratch.

The main functions used in any REST-based architecture are:

* **GET** − Provides read-only access to a resource.
* **PUT** − Creates a new resource.
* **DELETE** − Removes a resource.
* **POST** − Updates an existing resource or creates a new resource.

But all who claims cannot be referred to as RESTful API. In order to be regarded as a RESTful API, your application must satisfy certain constraints or principles. In the next section of this article on Building a REST API using Node.js, I will be talking about these principles in detail.

**Principles of REST**

Well, there are six ground principles laid down by Dr. Fielding who was the one to define the REST API design in 2000. Below are the six guiding principles of REST:

1. **Stateless**Requests sent from a client to the server contains all the necessary information that is required to completely understand it. It can be a part of the URI, query-string parameters, body, or even headers. The URI is used for uniquely identifying the resource and the body holds the state of the requesting resource. Once the processing is done by the server, an appropriate response is sent back to the client through headers, status or response body.
2. **Client-Server**

It has a uniform interface that separates the clients from the servers. Separating the concerns helps in improving the user interface’s portability across multiple platforms as well as enhance the scalability of the server components.

1. **Uniform Interface**

To obtain the uniformity throughout the application, REST has defined four interface constraints which are:

* + Resource identification
  + Resource Manipulation using representations
  + Self-descriptive messages
  + Hypermedia as the engine of application state

1. **Cacheable**

In order to provide a better performance, the applications are often made cacheable. It is done by labeling the response from the server as cacheable or non-cacheable either implicitly or explicitly. If the response is defined as cacheable, then the client cache can reuse the response data for equivalent responses in the future. It also helps in preventing the reuse of the stale data.

1. **Layered system**

The layered system architecture allows an application to be more stable by limiting component behavior.  This architecture enables load balancing and provides shared caches for promoting scalability. The layered architecture also helps in enhancing the application’s security as components in each layer cannot interact beyond the next immediate layer they are in.

1. **Code on demand**

Code on Demand is an optional constraint and is used the least. It permits a clients code or applets to be downloaded and extended via the interface to be used within the application. In essence, it simplifies the clients by creating a smart application which doesn’t rely on its own code structure.

Now that you know what is a REST API and what all you need to mind in order to deliver an efficient application, let’s dive deeper and see the process of building REST API using Node.js.

**Practical Demonstration: Building REST API using Node.js**

Here, we will be creating a simple CRUD REST application for Student Information using Node.js and Express.js. To build this application, you will need to install the following:

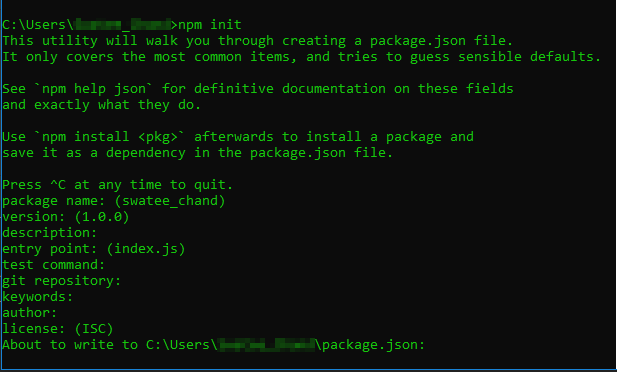
1. Node.js
2. Express.js
3. Joi
4. nodemon (Node Monitor)

In this example, I will be using the Notepad++ IDE to write and execute the codes. You can use any IDE or code editor according to your choice. So, let’s get started.

First, you need to create your project directory. Next, open the command prompt and navigate to your project directory. Once there, you need to call npm using the below command:

|  |  |
| --- | --- |
| 1 | * npm init |

When you hit enter, Node.js will ask you to enter some details to build the .json file such as:

Here you can define your entry point along with several other information. For this demo, I will be using **script.js** as an entry point.

Now, you need to install the required packages. In this project, I am using the below packages:

* **express.js:** It is a web framework.
* **mysql:** Node.js driver for MySQL
* **body-parser:** Helps in converting the POST data into the request body.
* **nodemon:** Helps in automatically restarting the server whenever the code changes.

In order to install these packages, type in the following command:

|  |  |
| --- | --- |
| 1 | * npm i --s express express-handlebars mongoose body-parser |

* npm install mysql

Finally, I will be installing a node monitoring package called **nodemon.** It keeps a watch on all the files with any type of extension present in this folder. Also, with nodemon on the watch, you don’t have to restart the Node.js server each time any changes are made. nodemon will implicitly detect the changes and restart the server for you.

Since I want to install nodemon such that it can access any file in the directory, I will be installing it with the global command:

|  |  |
| --- | --- |
| 1 | * npm i -g nodemon |