**API - Application program Interface**

**=============================**

Set of routines, protocols, tools to build software applications. basically, it tells how the software components should interacts.

Another way: it sits in between two applications and allows the both apps to establish the communication.

Example: Facebook messenger sends a message

**REST API**

**========**

A REST API is a web service API which uses URIs and HTTP protocol and JSON for data format.

A Uniform Resource Identifier (URI) is a unique sequence of characters that identifies a logical or physical resource used by web technologies.

The first thing to understand is that API is the superset while REST API is the subset. This means all REST APIs are APIs while not all APIs are REST APIs. API is a broad term while REST API is a specialized API.

While API is basically a set of functions and procedures that allow one application to access the feature of other application, REST is an architectural style for networked applications on the web. It is limited to client-server-based applications. REST is a set of rules or guidelines to build a web API.

REST - Representational State Transfer

http://myserver.com/catalog/item/1729 - Formatted REST API endpoint

http://myserver.com/catalog?item=1729 - Unformatted REST API endpoint

**Features of REST API**

**==================**

Stateless – Not all APIs are stateless, and REST API is which means that the call request to the API can be made individually and independently. Each call must have all the data necessary to execute the call. Moreover, it is due to its stateless property, the storage of cacheable data is possible. Hence, the speed of executing the calls and handling a large number of incoming calls can be done faster.

Interface – Not many APIs can decouple the client from the server or one application from another. REST API does it because of its uniform interface, and it allows the applications to evolve independently without tightly coupling the services, models, and actions to the API layer.

Scalability – A part of REST is indeed the URL part (it's the R in REST) but the is more important for scaling: state.

The server end of REST is stateless, which means that the server doesn't have to store anything across requests. therefore, less communication between servers, making it horizontally scalable.

Of course, there's a small bonus in the R (representational) in that a load balancer can easily route the request to the right server if you have nice URLs, and GET could go to a slave while POSTs go to masters.

Miscellaneous – REST APIs are the only few APIs that allow code on demand to be transmitted. One of the biggest advantages has to be the fact that the users do not have to know the function names and the parameters in a specific order to execute the REST APIs. It is independent of the type of platform or language you are using. It is highly reliable and absolutely simple and transparent in its execution.

**REST API in Action**

**================**

start spring with https://start.spring.io/

group name is the package name and the antifactory is the project name.

Select REST repository as a dependency for the REST application

After downloading unzip, the project and open in IntelliJ or Spring Tool Suite

Take a look at the pom.xml which is used to manage the dependencies. in spring boot app for example

locate the below

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

since this is a maven-based project, it will be the project dependency manager

which will download all the necessary .jar files.

take a look at the main application source code under src/main/java folder.

also notice the annotation - @SpringBootApplication

which tells the compiler that this is a spring boot application.

let’s create a new java class call HelloSR

add the below code.

package com.app1;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class heelosp {

@RequestMapping("/")

public String hello() {

return ("hello Welcome");

}

}

select the main class and run the app again

The Tomcat embed web server will enable to access over the browser as below

http://localhost:8080/

our REST API will give below output

welcome.

let’s further modify our API to a different url /abc.

@RequestMapping("/abc")

public String helloabc() {

return ("hello from abc");

}

your full REST API will looks like below

package com.app1;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class heelosp {

@RequestMapping("/")

public String hello() {

return ("hello Welcome");

}

@RequestMapping("/abc")

public String helloabc() {

return ("hello from abc");

}

}

run the app and test the new url as below

http://localhost:8080/abc