REST APIs

A PRACTICAL GUIDE

Key Elements of REST APIs - 1

Resources

- The first key element is the resource itself.
  - Let assume that a web application on a server has records of several employees.
  - Let's assume the URL of the web application is http://demo.guru99.com.
  - Now in order to access an employee record resource via REST, one can issue the command http://demo.guru99.com/employee/1
    - This command tells the web server to please provide the details of the employee whose employee number is 1.

Key Elements - 2

Request Verbs

These describe what you want to do with the resource. A browser issues a GET verb to instruct the endpoint it wants to get data. However, there are many other verbs available including things like POST, PUT, and DELETE. So in the case of the example http://demo.guru99.com/employee/1 , the web browser is actually issuing a GET Verb because it wants to get the details of the employee record.

Request Headers

These are additional instructions sent with the request. These might define the type of response required or the authorization details.

Request Body

Data is sent with the request. Data is normally sent in the request when a POST request is made to the REST web service. In a POST call, the client actually tells the web service that it wants to add a resource to the server. Hence, the request body would have the details of the resource which is required to be added to the server.

Response Body

This is the main body of the response. So in our example, if we were to query the web server via the request http://demo.guru99.com/employee/1 , the web server might return an XML document with all the details of the employee in the Response Body.

Response Status codes

These codes are the general codes which are returned along with the response from the web server. An example is the code 200 which is normally returned if there is no error when returning a response to the client.

Key Elements - 3

Restful Methods

The below list shows the verbs (POST, GET, PUT, and DELETE) and an example of what they would mean.

Let's assume that we have a RESTful web service is defined at the location. http://demo.guru99.com/employee . When the client makes any request to this web service, it can specify any of the normal HTTP verbs of GET, POST, DELETE and PUT.

Below is what would happen If the respective verbs were sent by the client.

POST – This would be used to create a new employee using the RESTful web service

GET - This would be used to get a list of all employee using the RESTful web service

PUT - This would be used to update all employee using the RESTful web service

DELETE - This would be used to delete all employee using the RESTful web service

An example

Let's take a look from a perspective of just a single record. Let's say there was an employee record with the employee number of 1.

The following actions would have their respective meanings.

POST – This would **not** be applicable since we are fetching data of employee 1 which is already created.

GET - This would be used to get the details of the employee with Employee no. as 1 using the RESTful web service

PUT - This would be used to update the details of the employee with Employee no. as 1 using the RESTful web service

DELETE - This is used to delete the details of the employee with Employee no. as 1

Why REST?

ADVANTAGES

Standardization

- Sort of … it's a loosely defined standard. No strict rules for syntax or behaviour. More guidelines

Simple interface representation

- As long as you can encode the data (usually json), you have your interface

Readable

- Again, json

Easy to CREATE APIs

- Human readable, no special tools (unlike SOAP)

DISADVANTAGES

Loose syntax and semantics

- Interface basically enforced in code

Often, a RESTful implementation is not (RESTful)

- There's no real way to enforce the principles

REST, to a large degree, owes it's growth to adoption by google, facebook, twitter … and it's simplicity

Network packets …

GET /api/v1/employee/1 HTTP/1.1
Host: dummy.restapiexample.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36 Edg/83.0.478.58
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://dummy.restapiexample.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=5a45cbef42e2fe9918eef269fc2862e6;

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Content-Type, X-Requested-With, X-authentication, X-client
Cache-Control: no-store, no-cache, must-revalidate
Content-Type: **application/json**;charset=utf-8
Date: Wed, 08 Jul 2020 17:09:12 GMT
Display: staticcontent_sol
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Host-Header: c2hhcmVkLmJsdWVob3N0LmNvbQ==
Pragma: no-cache
Referrer-Policy:
Response: 200
Server: nginx/1.16.0
Vary: Accept-Encoding
Vary: Accept-Encoding,User-Agent,Origin,X-APP-JSON
X-Ezoic-Cdn: Miss
X-Middleton-Display: staticcontent_sol
X-Middleton-Response: 200
X-Sol: pub_site
Content-Length: 134

{"status":"success","data":{"id":"1","employee_name":"Tiger Nixon","employee_salary":"320800","employee_age":"61","profile_image":""}}