NodeJs-Rest Apis

Agenda

- Introduction to REST basics.
- Stateless vs Statefull
- HTTP methods
- RESTFul Web Services
- Request and Response
- Building simple hello application with plain node.

What Is REST?

REST stands for REpresentational State Transfer.

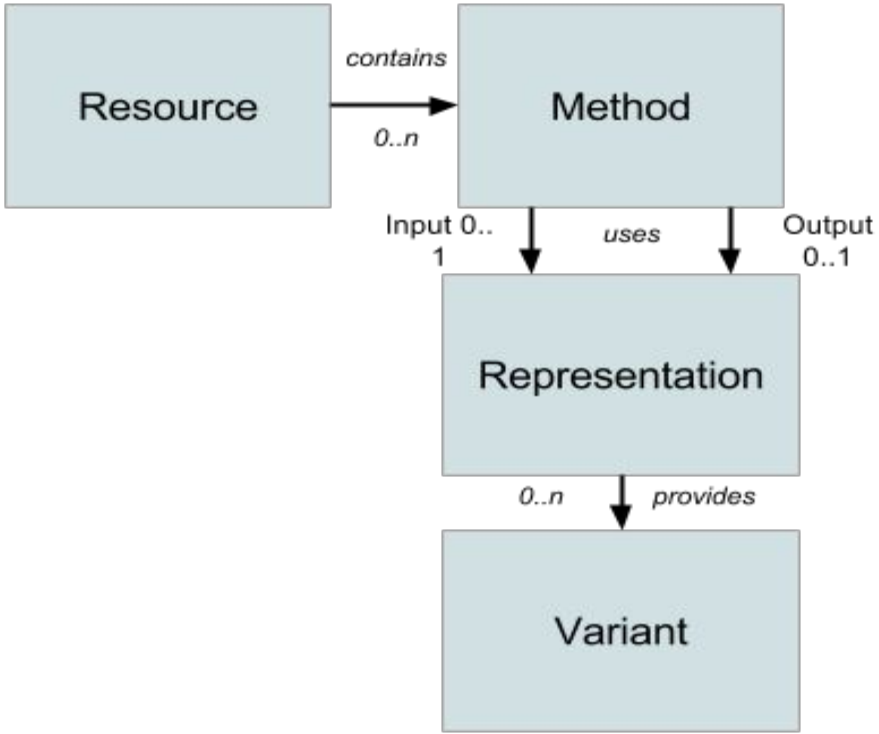
REST is web standards based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.

A REST Server simply provides access to resources and REST client accesses and modifies the resources using HTTP protocol.

Why REST

- Scalability
- General practice
- Technology Independence
- Security
- Encapsulation

Elements of REST Model



Resource

TO THE NEW DIGITAL

- The central element of REST
- A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.

A resource can be of many types:

- A resource to get the list of elements of this kind and to add new ones.
- Another to manage a particular element of this kind.
- And resources that manage no state and provide specific processing logic.

Methods

A predefined set of methods that get/update/delete a resource.

They are usually represented by a verb. eg:-

- GET
- POST
- PUT
- DELETE
- PATCH

Representation

- Structure of exchanged data
- Different Method use different representation of same resource
- For eg. Representation of PATCH might vary from PUT but both the call update the same resource.

Variant

Format in which data is exchanged. Example

- JSON
- XML
- YAML

Resource URL

A resource is attached to a URL in order to make it accessible from the Web.

This URL is known as Resource path.

They can be hierarchical, and this aspect can help us define relationships between managed entities.

There are several ways to send parameters to resources

- As a path variable:
  - /myentity/{entityid}
  - /user/{userid}/version/{versionid}

- As a query parameter:
  - what is specified after the ? within a URL

Stateless vs Statefull

- **A stateless** system can be seen as a box [black? ;)] where at any point in time the value of the output(s) depends only on the value of the input(s) [after a certain processing time]

- **A stateful** system instead can be seen as a box where at any point in time the value of the output(s) depends on the value of the input(s) and of an internal state, so basicaly a stateful system is like a state machine with "memory" as the same set of input(s) value can generate different output(s) depending on the previous input(s) received by the system.

HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

GET − This is used to provide a read only access to a resource.

POST − This is used to create a new resource.

DELETE − This is used to remove a resource.

PUT − This is used to update a existing resource or create a new resource.

HTTP methods - GET

- For retrieving the information
- Must be safe and idempotent
- For Collection Resource, /mobiles
  - It fetches all the element of that entity
- For Instance Resource, /mobile/{mobileId}
  - It fetches a single element of that resource

Note: An idempotent method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Safe method does not change the state or representation of resources

HTTP methods - GET

Can you guess the result for the following:

- /mobiles
- /mobiles/ab231
- /mobiles/ab231/variations

Complex Queries

- /mobiles?variant=iphone&screenSize=5

HTTP methods - DELETE

- For deleting an instance of resource identified by the URL
- DELETE operations are idempotent. Example:
  - DELETE http://www.example.com/customers/12345
  - DELETE http://www.example.com/customers/12345/orders
  - DELETE http://www.example.com/bucket/sample

HTTP methods - POST

- For creating a new Resource
- Can be used to create subordinate resources as well
- POST is neither safe nor idempotent.
- Making two identical POST requests will most-likely result in two resources containing the same information.

By convention, it should be ideally available to collection resources. Eg:-

- POST http://www.example.com/customers
  - {'firstName': “Prashant”, 'lastName': “Sharma”, “age”:25}

- POST http://www.example.com/customers/12345/orders
  - {'product':'ball', 'quantity': 2}

HTTP methods - PUT

- For updating resource mentioned in URL
- PUT is not a safe operation, in that it modifies state on the server, but it is idempotent
- By convention, we provide the complete representation of the resource in PUT with updated data

Examples

- PUT http://www.example.com/customers/12345
  - {'firstName': “Gaurav”, 'lastName': “Sharma”, “age”:29}

- PUT http://www.example.com/customers/12345/orders/98765
  - {'product':'ball', 'quantity': 5}

HTTP methods - PATCH

Same as PUT By convention, we provide only the updated information in the PATCH call. Eg

- PATCH http://www.example.com/customers/12345
  - {'firstName': “Gaurav”, “age”:29}
- PATCH http://www.example.com/customers/12345/orders/98765
  - {'quantity': 5}

RESTFul Web Services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., communication between Java and Python, or Windows and Linux applications) is due to the use of open standards.

Web services based on REST Architecture are known as RESTful web services. These webservices uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, which provides resource representation such as JSON and set of HTTP Methods.

Response

When a response comes back, there are two pieces of information to consider:

- **HTTP Status code** - this tells you at a protocol level what the status of the response is.

1. **Response Payload** - this is the body of the response. It is in a standard format that will give you the application level status, the results, and if applicable, any errors.

Response Codes

4xx Client Error
400 Bad Request
401 Unauthorized
403 Forbidden
404 Not Found

2xx Successful
200 success/OK
201 created
204 no content

5xx Server Error
500 Server Error
502 Bad Gateway
503 Unavailable

Response - POST

Collection URL, /customers

Success : 201 (Created)
Failure : 409 (Conflict)

Response - GET

Collection URL, /customers

Success : 200 (OK) [list of customers]
Instance URL, /customers/{id}
Success : 200 (OK) [single customer data]
Failure : 404 (Not Found) [Id not found or invalid]

Response - PUT

Collection URL, /customers

Failure : 404 (Not Found)
Instance URL, /customers/{id}
Success : 200 (OK), 204 (No Content)
Failure : 404 (Not Found) [Id not found or invalid]

Response - DELETE

Collection URL, /customers
Failure : 404 (Not Found)
Instance URL, /customers/{id}
Success : 200 (OK)
Failure : 404 (Not Found) [Id not found or invalid]

Request

The req object represents the HTTP**request** and has properties for the **request** query string, parameters, body, HTTP headers, and so on.

Following is the list of few properties associated with request object.

req.baseUrl:- The URL path on which a router instance was mounted.

req.body:- Contains key-value pairs of data submitted in the request body.

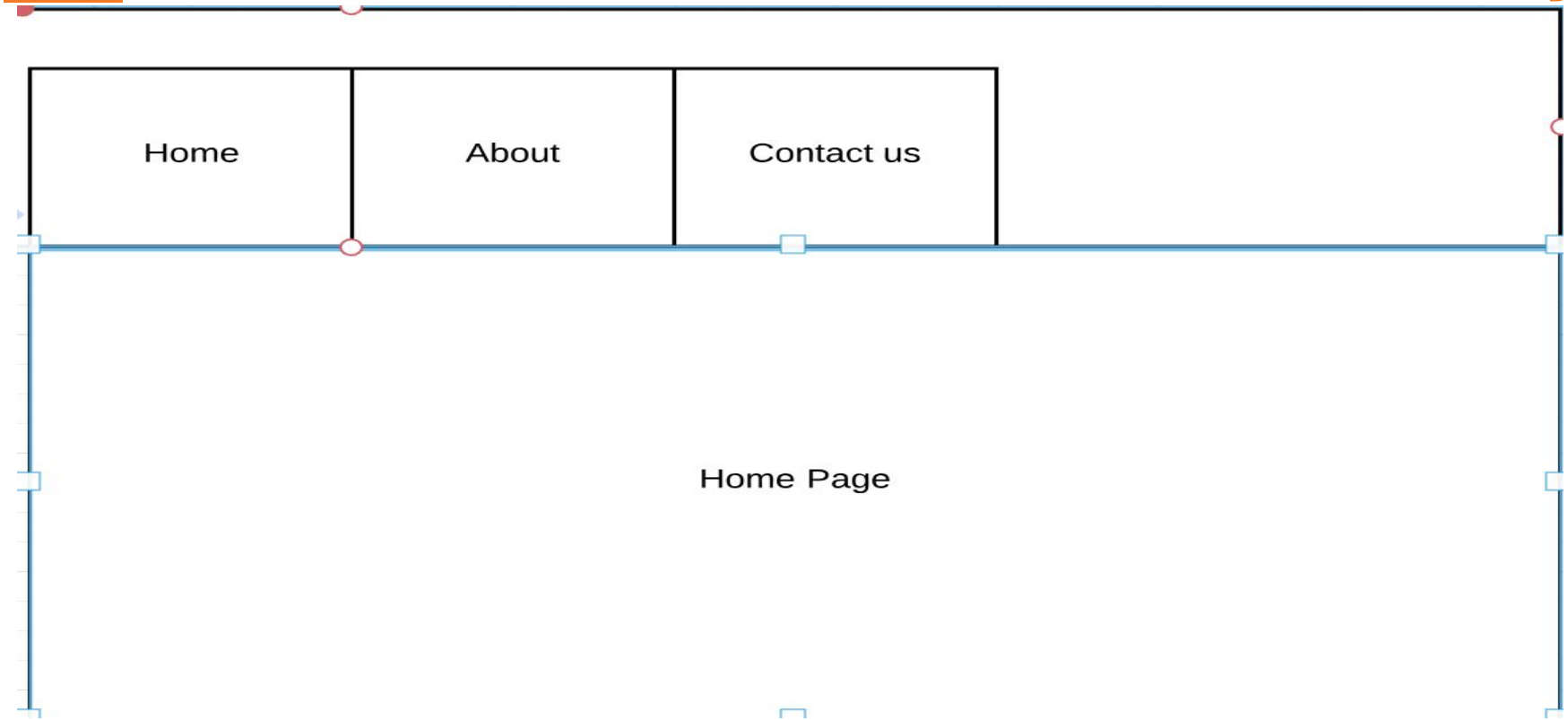
req.query:- An object containing a property for each query string parameter in the route.

Exercises

1. Create a REST API for User entity with following fields:
   Username
   Password
   FirstName
   LastName
   - Create search bar which will search by user name and will create an autocomplete list of user which fullfil the criteria.

2. Create a student API which will return list of all students.
   - Create a table which will show all users
   - Create filter for student branch name and on clicking on filters it will show students accordingly.
3. Create header which will show following items
   - Home
   - About
   - Contact us

   On clicking on the items it will load the pages


Home
About
Contact us
Home Page


THANK YOU