Rest Architectural Constraints

Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations

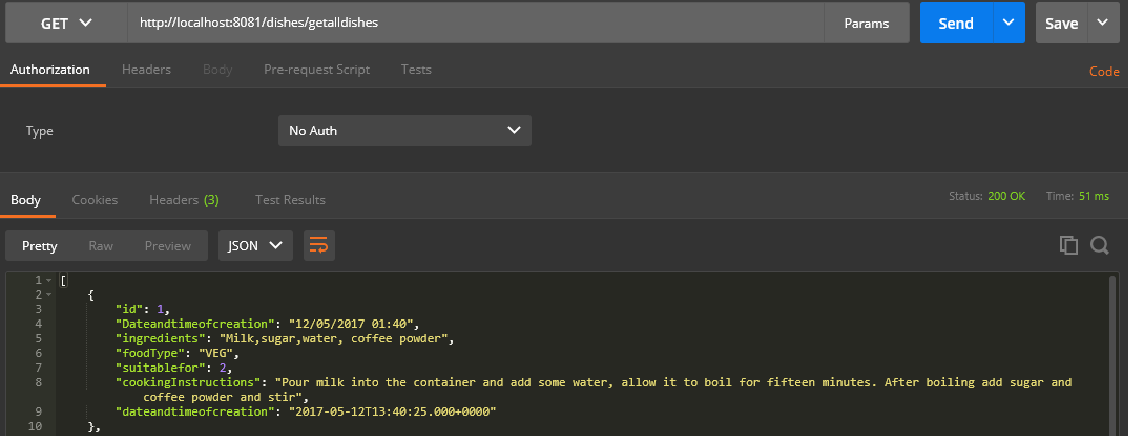
Let's demystify what that means (hopefully you got the full form). REST is basically a set of rules for communication between a client and server. There are a few constraints on the definition of REST:

1. Client-Server Architecture: the user interface of the website/app should be separated from the data request/storage, so each part can be scaled individually.
2. Statelessness: the communication should have no client context stored on server. This means each request to the server should be made with all the required data and no assumptions should be made if the server has any data from previous requests.
3. Layered system: client should not be able to tell if it is communicating directly with the server or some intermediary. These intermediary servers (be it proxy or load balancers) allow for scalability and security of the underlying server.

Some of the terms used in the heading:

1. REST Client: code or an app that can access these REST services. You are using one right now! Yes, the browser can act as an uncontrolled REST client (the website handles the browser requests). The browser, for a long time, used an in-built function called XML Http Request for all REST requests. But, this was succeeded by FetchAPI, a modern, [promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise)based approach to requests. Other examples are code libraries like axios, superagent and [got](https://github.com/sindresorhus/got) or some dedicated apps like [Postman](https://www.postman.com/).
2. REST Service: REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ global IDs. REST uses various representation to represent a resource like text, JSON, XML.
3. REST API: this defines the endpoint and methods allowed to access/submit data to the server. We will talk about this in great detail below. Other alternatives to this are: GraphQL, JSON-Pure and oData.

How Rest Looks Like  
In programming terms, there is an endpoint (a URL) that the server is waiting to get a request. We connect to that endpoint and send in some data about us (remember, REST is stateless, no data about the request is stored) and the server responds with the correct response.



The returned data is in JSON (JavaScript Object Notation) and can be accessed directly. Here, **http://localhost:8081/dishes/getalldishes** is called an endpoint of an API. There will be a server listening on that endpoint for requests like the one we made.

## Anatomy of REST:

Alright, so now we know that data can be requested by the client and the server will respond appropriately. Let's look deeper into how a request is formed.

1. Endpoint: I have already told you about this. For a refresher, it is the URL where the REST Server is listening.
2. Method: Earlier, I wrote that you can either request data or modify it, but how will the server know what kind of operation the client wants to perform? REST implements multiple 'methods' for different types of requests, the following are most popular:

* GET: Get resource from the server.
* POST: Create resource to the server.
* PATCHor PUT: Update existing resource on the server.
* DELETE: Delete existing resource from the server.

1. **Headers:** The additional details provided for communication between client and server (remember, REST is stateless). Some of the common headers are:

**Request:**

* host: the IP of client (or from where request originated)
* accept-language: language understandable by the client
* user-agent: data about client, operating system, and vendor

**Response:**

* status: the status of request or HTTP code.
* content-type: type of resource sent by server.
* set cookie sets cookies by server

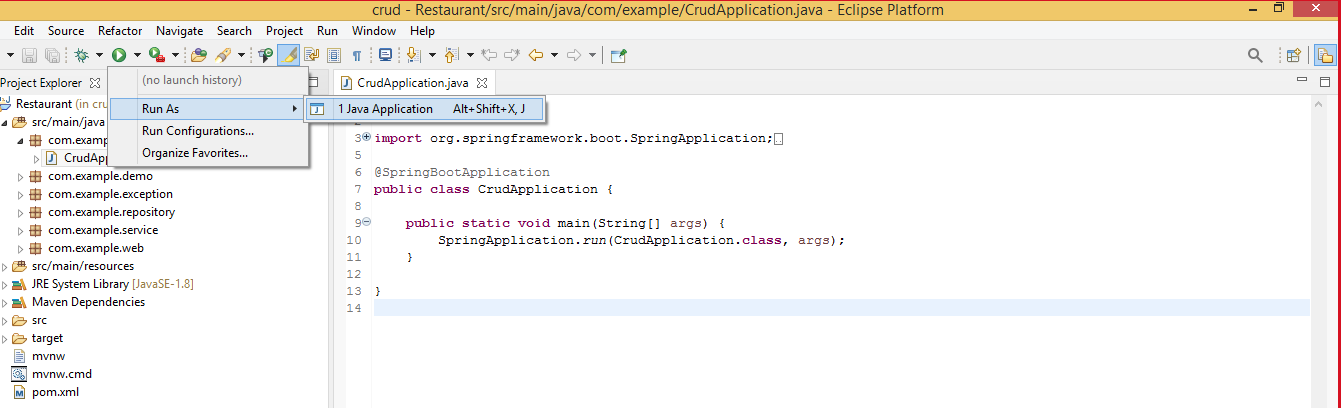
1. **Data:** (also called body or message) contains info you want to send to the server.

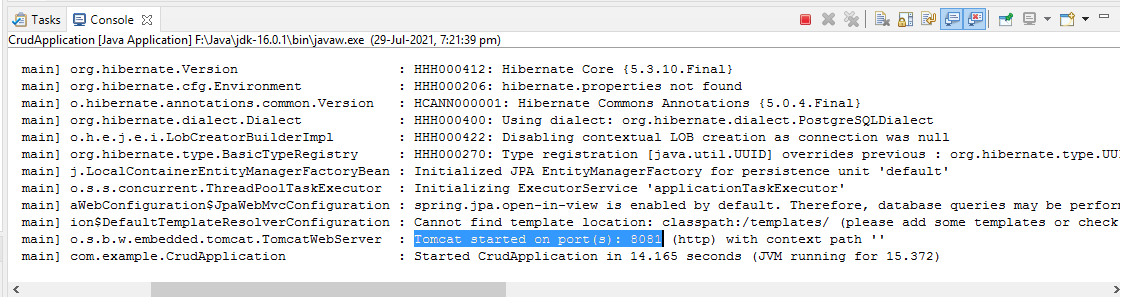
Web application which allows users to manage your favourite recipes

First, we need to create database connection, please refer to attached file restaurantdatabase. We have used PostgreSQL for database connection.

How to run the application

Step 1:Extract the zip file and import the project using eclipse. Run the application. Internally tomcat server runs in the eclipse.

****

****

Step 2:Open postman tool**,** We will be using postman to show the request and response, this postman tool acts as a client.

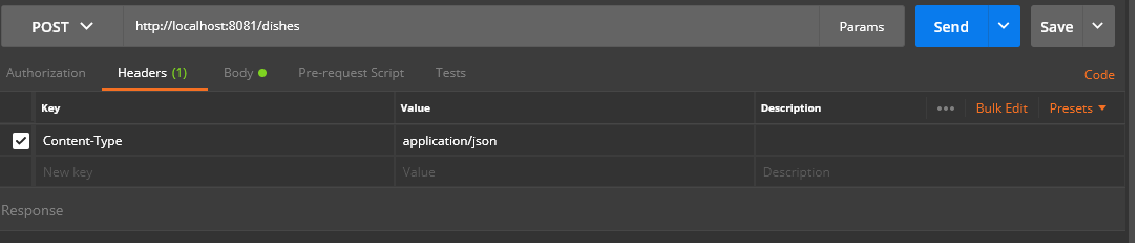
Step 3**:** Create a recipe by using POST method**.**

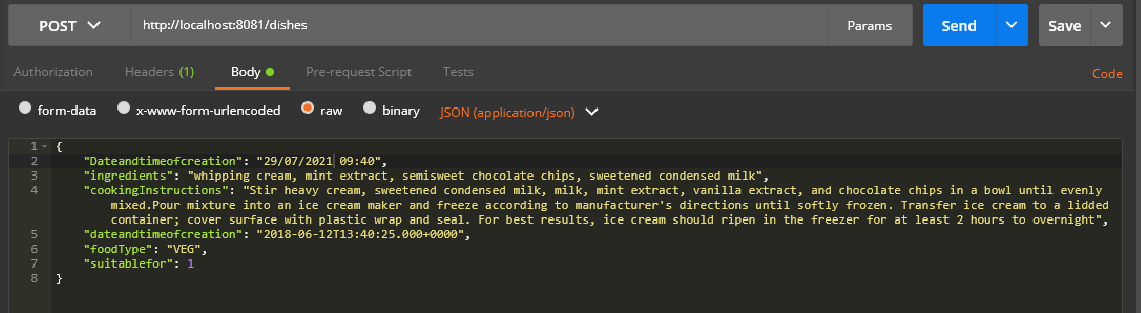
URL  **:** http://localhost:8081/dishes

Headers **:** "key":"Content-Type",

"value" :"application/json"

Body **:** raw -> select JSON(application/json)





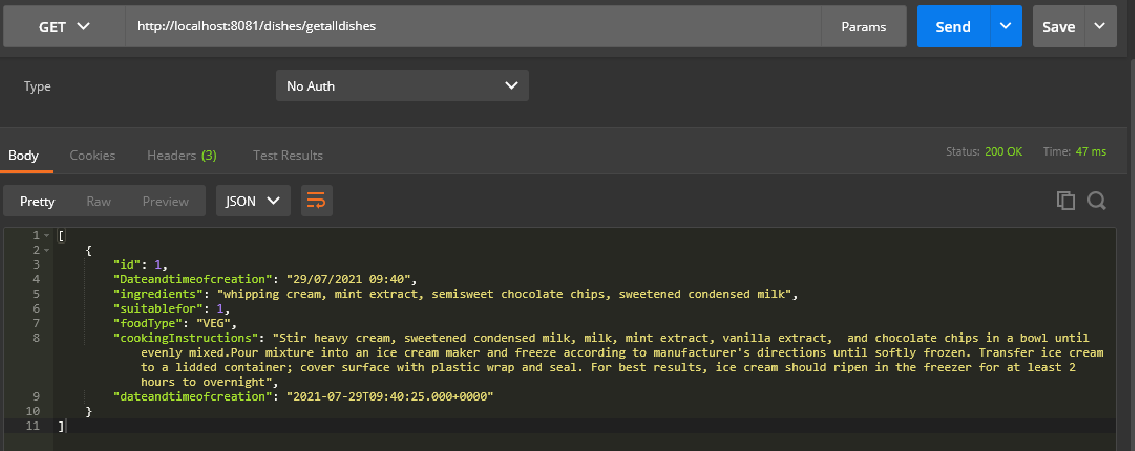
Insert the JSON formatted code in the body

|  |
| --- |
| {  "Dateandtimeofcreation": "29/07/2021 09:40",  "ingredients": "whipping cream, mint extract, semisweet chocolate chips, sweetened condensed milk",  "cookingInstructions": "Stir heavy cream, sweetened condensed milk, milk, mint extract, vanilla extract, and chocolate chips in a bowl until evenly mixed. Pour mixture into an ice cream maker and freeze according to manufacturer's directions until softly frozen. Transfer ice cream to a lidded container; cover surface with plastic wrap and seal. For best results, ice cream should ripen in the freezer for at least 2 hours to overnight",  "dateandtimeofcreation": "2018-06-12T13:40:25.000+0000",  "foodType": "VEG",  "suitablefor": 1  } |

Step 4:To get recipe from restaurant menu by using GET method**.**

URL :http://localhost:8081/dishes/getalldishes

No header or body is required for GET method.

****

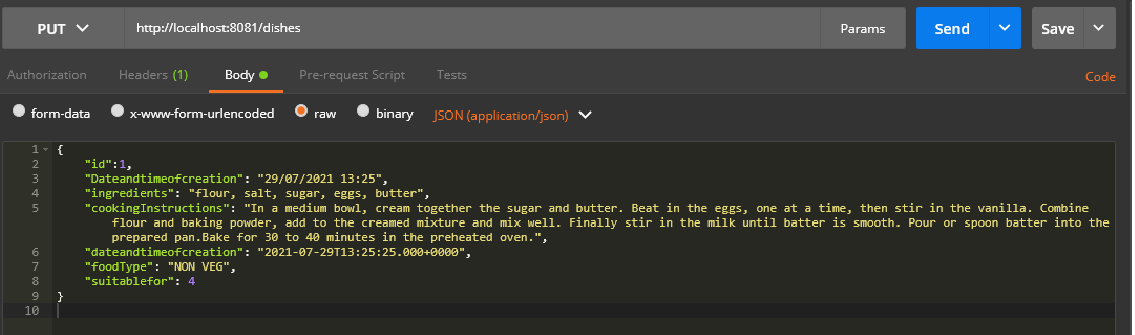
Step 5:To update a particular recipe from restaurant menu by using PUT method**.**

URL :http://localhost:8081/dishes

Headers**:** "key":"Content-Type",

"value": "application/json"

Body :raw -> select JSON(application/json)

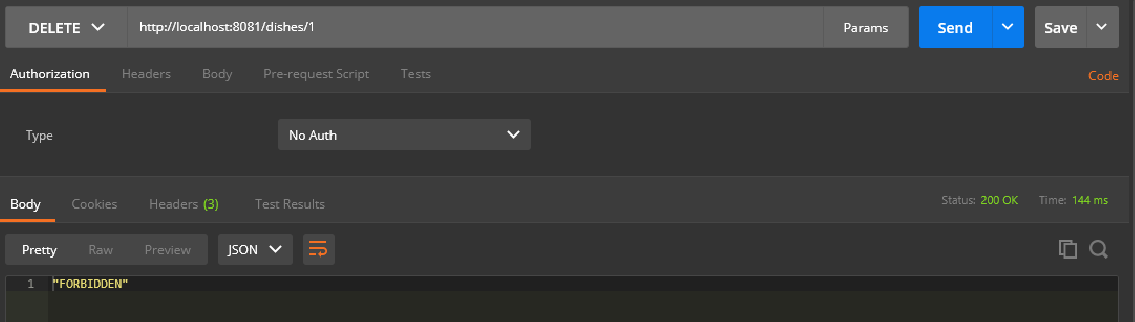


Update the JSON formatted code in the body

|  |
| --- |
| {  "id":1,  "Dateandtimeofcreation": "29/07/2021 13:25",  "ingredients": "flour, salt, sugar, eggs, butter",  "cookingInstructions": "In a medium bowl, cream together the sugar and butter. Beat in the eggs, one at a time, then stir in the vanilla. Combine flour and baking powder, add to the creamed mixture and mix well. Finally stir in the milk until batter is smooth. Pour or spoon batter into the prepared pan.Bake for 30 to 40 minutes in the preheated oven.",  "dateandtimeofcreation": "2021-07-29T13:25:25.000+0000",  "foodType": "NON VEG",  "suitablefor": 4  } |

Step 6 :To delete a particular receipe from restaurant menu by using DELETE method with ID**.**

URL :http://localhost:8081/dishes/1

****