Initially, we need to define the employee entity. Therefore, the following employee class is defined:

public class Employee {

public Employee() {}

public Employee(

Integer id, String firstName,

String lastName, String email)

{

super();

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

}

private Integer id;

private String firstName;

private String lastName;

private String email;

@Override

public String toString()

{

return "Employee [id="

+ id + ", firstName="

+ firstName + ", lastName="

+ lastName + ", email="

+ email + "]";

}

public Integer getId()

{

return id;

}

public void setId(Integer id)

{

this.id = id;

}

public String getFirstName()

{

return firstName;

}

public void setFirstName(

String firstName)

{

this.firstName = firstName;

}

public String getLastName()

{

return lastName;

}

public void setLastName(

String lastName)

{

this.lastName = lastName;

}

public String getEmail()

{

return email;

}

public void setEmail(String email)

{

this.email = email;

}

}

Now, we need to create a storage class which stores the list of all the employees:

import java.util.ArrayList;

import java.util.List;

public class Employees {

private List<Employee> employeeList;

public List<Employee> getEmployeeList()

{

if (employeeList == null) {

employeeList

= new ArrayList<>();

}

return employeeList;

}

public void

setEmployeeList(

List<Employee> employeeList)

{

this.employeeList

= employeeList;

}

}

Till now, we have defined the entity employee and created a storage class. Now, we need to access the employees. So, we create a class from where we will create an object of the storage class to store the employees:

import org.springframework

.stereotype

.Repository;

import com.example.demo.Employees;

@Repository

public class EmployeeDAO {

private static Employees list

= new Employees();

static

{

list.getEmployeeList().add(

new Employee(

1,

"Prem",

"Tiwari",

"chapradreams@gmail.com"));

list.getEmployeeList().add(

new Employee(

2, "Vikash",

"Kumar",

"abc@gmail.com"));

list.getEmployeeList().add(

new Employee(

3, "Ritesh",

"Ojha",

"asdjf@gmail.com"));

}

public Employees getAllEmployees()

{

return list;

}

public void

addEmployee(Employee employee)

{

list.getEmployeeList()

.add(employee);

}

}

Finally, we need to create a controller class which is the actual implementation of the REST API. According to the REST rules, every new entry in the database has to be called by the POST method and all the requests from the database must be called using the GET method. The same methods are implemented in the following code:

package com.example.demo;

import java.net.URI;

import org.springframework.beans

.factory.annotation.Autowired;

import org.springframework.http

.ResponseEntity;

import org.springframework.web.bind

.annotation.GetMapping;

import org.springframework.web.bind

.annotation.PostMapping;

import org.springframework.web.bind

.annotation.RequestBody;

import org.springframework.web.bind

.annotation.RequestMapping;

import org.springframework.web.bind

.annotation.RestController;

import org.springframework.web.servlet

.support.ServletUriComponentsBuilder;

import com.example.demo.Employees;

import com.example.demo.EmployeeDAO;

import com.example.demo.Employee;

@RestController

@RequestMapping(path = "/employees")

public class EmployeeController {

@Autowired

private EmployeeDAO employeeDao;

@GetMapping(

path = "/",

produces = "application/json")

public Employees getEmployees()

{

return employeeDao

.getAllEmployees();

}

@PostMapping(

path = "/",

consumes = "application/json",

produces = "application/json")

public ResponseEntity<Object> addEmployee(

@RequestBody Employee employee)

{

Integer id

= employeeDao

.getAllEmployees()

.getEmployeeList()

.size()

+ 1;

employee.setId(id);

employeeDao

.addEmployee(employee);

URI location

= ServletUriComponentsBuilder

.fromCurrentRequest()

.path("/{id}")

.buildAndExpand(

employee.getId())

.toUri();

return ResponseEntity

.created(location)

.build();

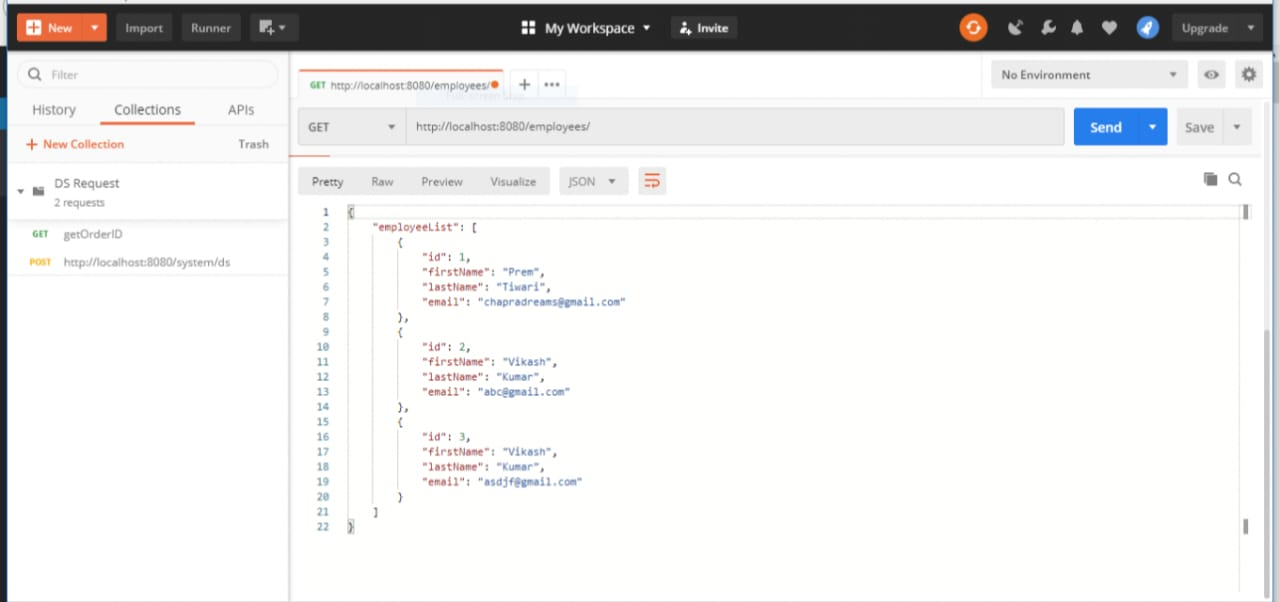
}

}

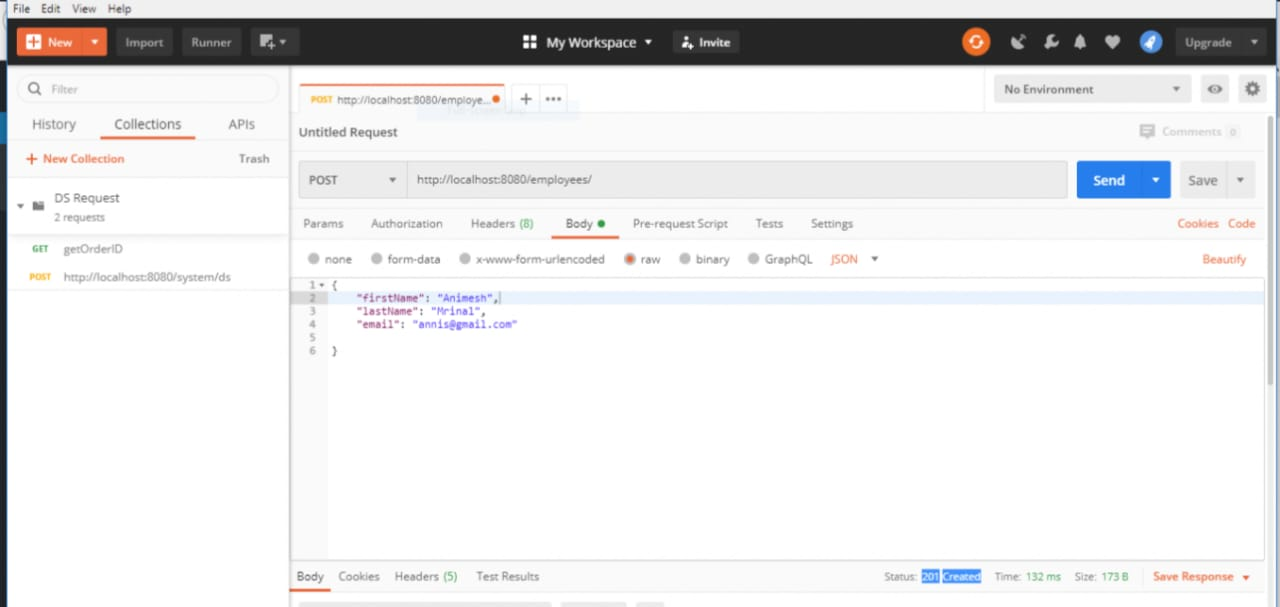
After implementing all the classes in the project, run the project as **Spring Boot App.** Once the server starts running, we can send the requests through the browser or postman. We can access the running app by going into the following URL

**Output:**

When a GET request is performed:

****

When a POST request is performed:

****

Again hitting the GET request after performing the POST request:

