Load following json data into spring boot application and persist into any database

[

{"id":1,"first\_name":"Padget","last\_name":"Wannop","email":"pwannop0@csmonitor.com","gender":"Male","departments":[{"department\_id":1,"department\_name":"Marketing","department\_location":"Kazakhstan"},{"department\_id":2,"department\_name":"Services","department\_location":"Brazil"}],"dob":"4/20/1968"},

{"id":2,"first\_name":"Andris","last\_name":"Mance","email":"amance1@edublogs.org","gender":"Male","departments":[{"department\_id":1,"department\_name":"Marketing","department\_location":"Russia"},{"department\_id":2,"department\_name":"Product Management","department\_location":"Greece"},{"department\_id":3,"department\_name":"Research and Development","department\_location":"Russia"},{"department\_id":4,"department\_name":"Services","department\_location":"France"},{"department\_id":5,"department\_name":"Business Development","department\_location":"Brazil"}],"dob":"12/1/1963"}

]

To create a Spring Boot application that loads JSON array objects for the given data, you can follow these steps:

1. Create a new Spring Boot project using the Spring Initializer or your preferred IDE.
2. Create a POJO (Plain Old Java Object) class to represent the JSON objects in your data. For example, you can create a class called Employee with the following properties: id, first\_name, last\_name, email, gender, departments (as a List of Department objects), and dob.
3. Create another POJO class called Department to represent the nested department objects in the JSON data. This class should have the following properties: department\_id, department\_name, and department\_location.
4. In your Spring Boot application, create a REST controller class that will handle the incoming JSON data. In this class, use the @RestController annotation to indicate that it is a REST controller.
5. Create a method in the controller class that will handle the incoming JSON data. Use the @PostMapping annotation to indicate that this method will handle POST requests. The method should take in a List of Employee objects as a parameter.
6. In the method, you can use the Jackson ObjectMapper to convert the incoming JSON data into a List of Employee objects.
7. Once the JSON data has been converted, you can use the data as needed in your application. For example, you could save it to a database or perform some operations on it.
8. Finally, you can test your application by sending a POST request to the endpoint with the JSON data as the request body.

Note: The above steps provide a general overview of how to create a Spring Boot application that loads JSON array objects. Depending on your specific requirements, there may be additional steps or modifications needed.

Top of Form

import java.util.List;

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private String email;

private String gender;

private List<Department> departments;

private String dob;

//getters and setters

}

public class Department {

private int department\_id;

private String department\_name;

private String department\_location;

//getters and setters

}

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

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

import java.util.List;

@RestController

public class EmployeeController {

@PostMapping("/employees")

public void loadEmployees(List<Employee> employees) {

// do something with the employees

}

}

import com.fasterxml.jackson.databind.ObjectMapper;

import java.util.List;

@RestController

public class EmployeeController {

@PostMapping("/employees")

public void loadEmployees(String json) {

ObjectMapper objectMapper = new ObjectMapper();

try {

List<Employee> employees = objectMapper.readValue(json, new TypeReference<List<Employee>>(){});

// do something with the employees

} catch (IOException e) {

e.printStackTr

Bottom of Form