**Exercise 3: Employee Management System - Creating Repositories**

* **Overview of Spring Data Repositories:**

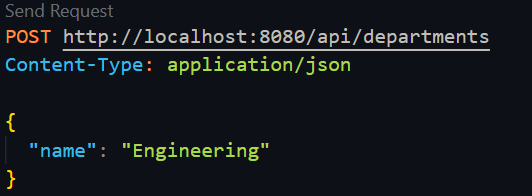
1. **Simplified Data Access**: Spring Data repositories provide a convenient way to implement data access layers by extending interfaces, eliminating the need for boilerplate code and reducing development time.
2. **Built-in CRUD Methods**: By extending *JpaRepository*, you gain access to built-in CRUD methods such as *save*, *findAll, findById, deleteById,* etc., simplifying common database operations.
3. **Type Safety and Flexibility**: Repositories provide type safety and flexibility through generics, allowing operations on specific entity types without casting or errors at runtime.
4. **Derived Query Methods**: Spring Data allows you to define query methods based on entity attributes, automatically generating the required SQL queries based on method names.

* **Creating Repositories:**

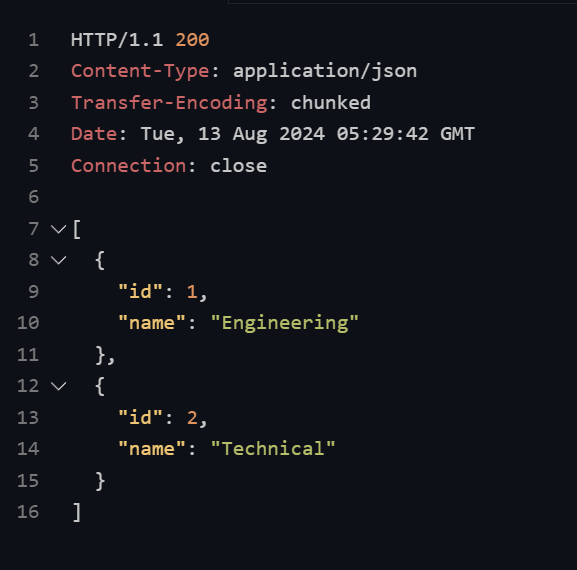
1. ***EmployeeRepository* Interface**: Create an *EmployeeRepository* interface by extending *JpaRepository<Employee, Long>,* providing CRUD operations for Employee entities with Long IDs.
2. ***DepartmentRepository* Interface**: Create a *DepartmentRepository* interface by extending *JpaRepository<Department, Long>,* enabling CRUD operations for Department entities with Long IDs.
3. **Define Query Methods**: Define derived query methods in both repositories, such as *findByLastName(String lastName)* in *EmployeeRepository*, to retrieve data based on specific attributes.
4. **Automatic Implementation**: Spring automatically implements these repositories, allowing the application to inject and use them without manual implementation, thanks to Spring's dependency injection and proxy mechanisms.

* **OutPut of the code :**

To add and update employees and departments using a REST client. A REST client is a tool or application used to interact with RESTful web services by sending HTTP requests and receiving responses. It allows developers to test and debug APIs by performing CRUD operations (Create, Read, Update, Delete) on resources over the web.

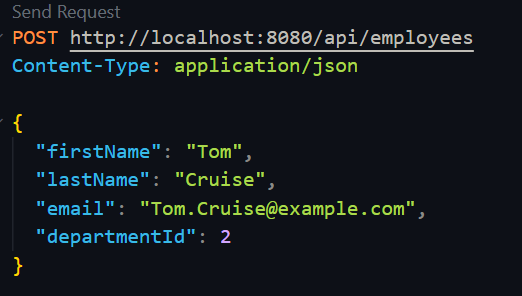
* ****Add department:** Use the **POST** method and send a request to the endpoint ***http://localhost:8080/api/departments.***

In this way add the dept.

 This is the REST Client response of adding departments.

* **Add an Employee:** Use the **POST** method to send a request to ***http://localhost:8080/api/employees.***

add Employees in this way

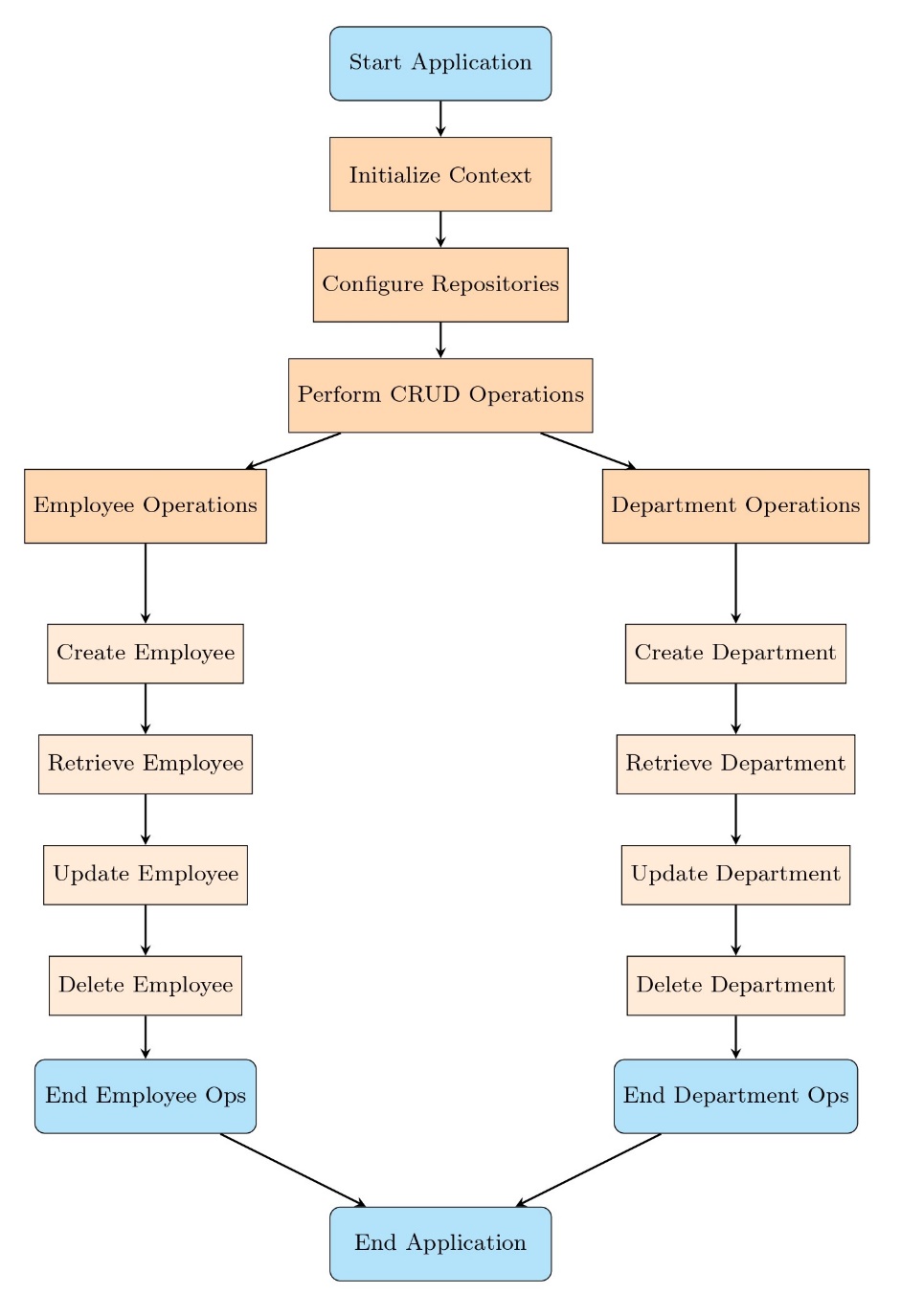


This is the all-Employee list ( response from the REST CLIENT )

**The GitHub link of the code -** [link](https://github.com/Hyperstrom/Aniket-Pal_5017587/tree/main/WEEK-3/Exercise-3)

* **The Flow-Chart of the program:**

1. ***Start and Initialization:*** The application starts and initializes the Spring application context, which configures and loads all necessary beans.
2. ***Repository Configuration***: The EmployeeRepository and DepartmentRepository are configured and made available for dependency injection through Spring's automatic implementation mechanism.
3. ***Perform CRUD Operations:*** The application performs CRUD operations on Employee and Department entities using the repositories, leveraging built-in methods and any defined derived queries.



1. ***End Application:*** Once the operations are complete, the application can shut down gracefully, releasing resources.