**Employee Management System - Enabling Entity Auditing**

* To solve this problem auditing was implemented for the Employee and Department entities in the Employee Management System.
* **The goal was to track the creation and modification dates of these entities, allowing us to monitor when records were created and last updated.**
* To do this, we leveraged Spring Data JPA's auditing capabilities, combined with Lombok to reduce boilerplate code.

**Key Components and Annotations:**

1. **Auditable Class:**
   * **Purpose:** It is a base class which encapsulates the common auditing fields (createdDate and lastModifiedDate) and their related behavior.
   * **Annotations:**
     + **@MappedSuperclass:** Indicates that this class is a base class whose properties should be inherited by child entities.
     + **@EntityListeners(AuditingEntityListener.class):** Specifies that the Auditable class will use the AuditingEntityListener to automatically populate the auditing fields.
     + **@CreatedDate:** Marks the field to be automatically populated with the creation timestamp when the entity is first persisted.
     + **@LastModifiedDate:** Marks the field to be automatically updated with the timestamp whenever the entity is updated.
   * **Lombok Annotations:**
     + **@Getter** and **@Setter:** Automatically generate getter and setter methods for the fields, reducing boilerplate code.
2. **Employee and Department Entities:**
   * **Inheritance:** Both Employee and Department entities inherit from the Auditable class, which means they automatically have the auditing fields (createdDate and lastModifiedDate) without the need to declare them again.
   * **Lombok Integration:** The **@Data** annotation on both entities is used to automatically generate all necessary boilerplate code like **getters, setters, toString, equals,** and **hashCode** methods. This makes the entities more concise and easier to maintain.
3. **Enabling JPA Auditing:**
   * **@EnableJpaAuditing** annotation is added to the main application class (EmployeeManagementSystemApplication) to activate the auditing feature in Spring Data JPA. Without this annotation, the auditing fields would not be populated automatically.
4. **Application Properties:**
   * **JPA Configuration:** The **spring.jpa.show-sql=true** property is included to display the SQL queries generated by JPA in the console, helping to verify that auditing is working as expected.

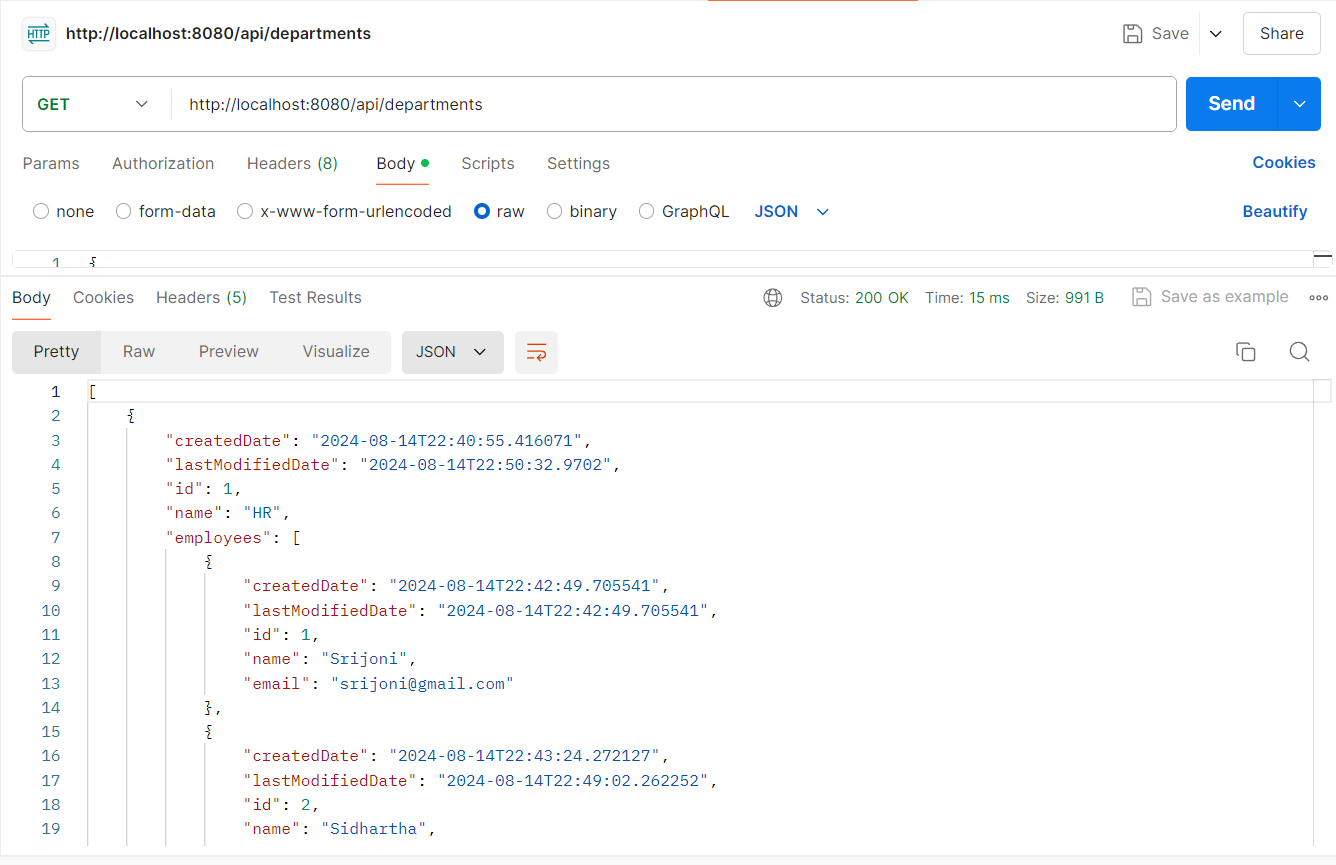
**Workflow Explanation:**

1. **Creating Entities:**
   * When a new Employee or Department entity is created and saved, the **@CreatedDate** field is automatically populated with the current timestamp.
2. **Updating Entities:**
   * When an existing Employee or Department entity is updated, the **@LastModifiedDate** field is automatically updated with the current timestamp, reflecting the last modification time.
3. **Data Consistency and Integrity:**
   * The auditing fields ensure that every record has consistent metadata about its lifecycle, which is crucial for tracking changes, debugging, and maintaining data integrity in the system.

**Benefits of Using Lombok and Auditing:**

* **Reduced Boilerplate Code:** Lombok annotations **(@Data, @Getter, @Setter**) eliminate the need for manually writing getters, setters, and other utility methods, making the codebase cleaner and easier to maintain.
* **Automatic Auditing:** The combination of **@CreatedDate** and **@LastModifiedDate** annotations with Spring Data JPA's auditing features allows for seamless tracking of entity lifecycle events without additional manual intervention.
* **Scalability:** By centralizing the auditing fields in the Auditable base class, any new entity that requires auditing can simply extend this class, ensuring consistency across the application.

This feature is particularly useful in enterprise applications where data tracking and history are critical for compliance, auditing, and debugging purposes.

* **Testing of the Auditing:**