**Exercise 9: Employee Management System - Customizing Data Source Configuration**

Business Scenario:

Customize your data source configuration and manage multiple data sources.

**1. Introduction**

The Employee Management System (EMS) is designed to manage employee and department data using a Spring Boot application. It includes functionalities for CRUD operations, managing multiple data sources, and configuring JPA auditing.

**2. Key Components**

**2.1. Entities**

* **Department**:
  + **Fields**:
    - id: Unique identifier for the department (Long).
    - name: Name of the department (String).
    - createdDate: Date when the department was created (Date).
    - lastModifiedDate: Date when the department was last modified (Date).
  + **Relationships**:
    - One-to-Many relationship with Employee.
* **Employee**:
  + **Fields**:
    - id: Unique identifier for the employee (Long).
    - name: Name of the employee (String).
    - email: Email of the employee (String).
    - createdDate: Date when the employee was created (Date).
    - lastModifiedDate: Date when the employee was last modified (Date).
  + **Relationships**:
    - Many-to-One relationship with Department.

**2.2. Repositories**

* **DepartmentRepository**:
  + **Methods**:
    - save(Department department): Saves a department entity.
    - findAll(): Retrieves all department entities.
    - findById(Long id): Finds a department by its ID.
    - deleteById(Long id): Deletes a department by its ID.
* **EmployeeRepository**:
  + **Methods**:
    - save(Employee employee): Saves an employee entity.
    - findAll(): Retrieves all employee entities.
    - findByName(String name): Finds employees by name.
    - findByEmail(String email): Finds an employee by email.
    - findById(Long id): Finds an employee by its ID.
    - deleteById(Long id): Deletes an employee by its ID.
* **EmployeeProjection**:
  + **Methods**:
    - getId(): Retrieves the employee ID.
    - getName(): Retrieves the employee name.
    - getEmail(): Retrieves the employee email.

**2.3. Data Source Configurations**

* **PrimaryDataSourceConfig**:
  + **Methods**:
    - primaryDataSource(): Configures and returns the primary data source.
    - primaryEntityManagerFactory(DataSource dataSource): Configures and returns the entity manager factory for the primary data source.
    - primaryTransactionManager(EntityManagerFactory entityManagerFactory): Configures and returns the transaction manager for the primary data source.
* **SecondaryDataSourceConfig**:
  + **Methods**:
    - secondaryDataSource(): Configures and returns the secondary data source.
    - secondaryEntityManagerFactory(DataSource dataSource): Configures and returns the entity manager factory for the secondary data source.
    - secondaryTransactionManager(EntityManagerFactory entityManagerFactory): Configures and returns the transaction manager for the secondary data source.

**2.4. Controllers**

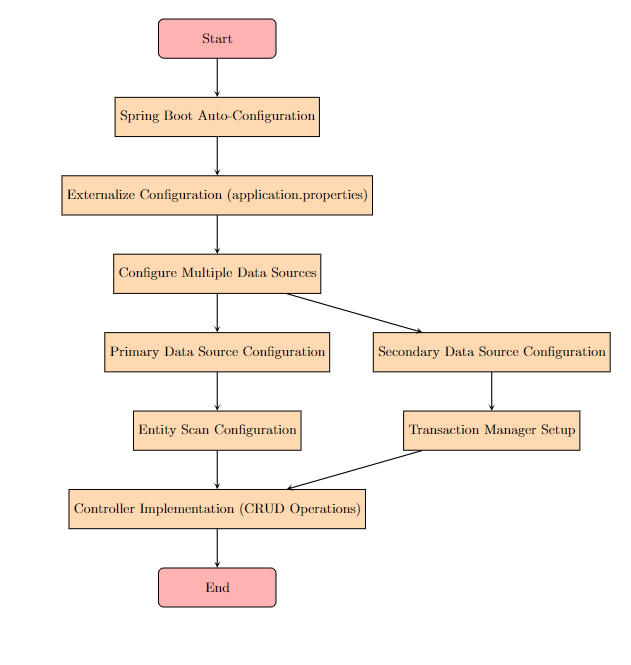
* **DepartmentController**:
  + **Endpoints**:
    - GET /departments: Retrieves all departments.
    - POST /departments: Creates a new department.
    - GET /departments/{id}: Retrieves a department by its ID.
    - PUT /departments/{id}: Updates a department by its ID.
    - DELETE /departments/{id}: Deletes a department by its ID.
* **EmployeeController**:
  + **Endpoints**:
    - GET /employees: Retrieves employees with pagination.
    - GET /employees/sorted: Retrieves employees with sorting and pagination.
    - POST /employees: Creates a new employee.
    - GET /employees/{id}: Retrieves an employee by its ID.
    - PUT /employees/{id}: Updates an employee by its ID.
    - DELETE /employees/{id}: Deletes an employee by its ID.

**3. Data Source Configuration**

* **Primary Data Source**:
  + Configured with H2 database for in-memory operations.
* **Secondary Data Source**:
  + Configured with MySQL database for persistent storage.

**4. JPA Auditing**

* Enabled using @EnableJpaAuditing in EmployeemanagementsystemApplication.
* Annotated Department and Employee entities with @CreatedDate and @LastModifiedDate to track creation and modification times.

**FLOWCHART:**

**Explanation:**

* **Start**: The process begins.
* **Spring Boot Auto-Configuration**: Leverage Spring Boot's auto-configuration to set up the default data source configurations.
* **Externalize Configuration (application.properties)**: Move data source configurations to the application.properties file to externalize and manage settings.
* **Configure Multiple Data Sources**: Set up configurations for both primary and secondary data sources.
* **Primary Data Source Configuration**: Configure the primary data source, including the connection properties and entity scanning.
* **Secondary Data Source Configuration**: Similarly, configure the secondary data source with its own settings.
* **Entity Scan Configuration**: Specify which packages contain the JPA entities to be managed by each data source.
* **Transaction Manager Setup**: Configure transaction management for both data sources to ensure proper handling of transactions.
* **Controller Implementation (CRUD Operations)**: Implement controllers to handle CRUD operations on the entities, using the configured data sources.
* **End**: The process concludes.

**CLASS DIAGRAM :**



**Explanation:**

1. **Entities**:
   * **Department**: Represents a department with fields for id, name, createdDate, and lastModifiedDate.
   * **Employee**: Represents an employee with fields for id, name, email, createdDate, and lastModifiedDate. It has a many-to-one relationship with Department.
2. **Repositories**:
   * **DepartmentRepository**: Interface for managing Department entities with methods for CRUD operations.
   * **EmployeeRepository**: Interface for managing Employee entities with methods for CRUD operations and custom queries.
   * **EmployeeProjection**: Interface for fetching specific data from the Employee entity.
3. **Data Source Configurations**:
   * **PrimaryDataSourceConfig**: Configuration class for the primary data source, including methods for creating the data source, entity manager factory, and transaction manager.
   * **SecondaryDataSourceConfig**: Similar to PrimaryDataSourceConfig, but for the secondary data source.
4. **Controllers**:
   * **DepartmentController**: Handles HTTP requests related to Department entities, such as getting all departments, creating, updating, and deleting a department.
   * **EmployeeController**: Handles HTTP requests related to Employee entities, including operations with pagination and sorting.