**Employee Management System - Customizing Data Source Configuration**

**Overview**

In this exercise, the focus is on customizing data source configuration to manage multiple data sources within the application.

**Objective**

The objective of this exercise is to:

1. Leverage Spring Boot's auto-configuration capabilities for data sources.
2. Externalize configuration settings using the application.properties file.
3. Manage multiple data sources, ensuring that the Employee and Department entities are stored in separate databases.

**Project Structure**

The project is organized as follows:

EmployeeManagementSystem

* src/main/java

1. com.code.department.entity

* Department.java

1. com.code.department.repositories

* DepartmentRepository.java

1. com.code.employee

* EmployeeProjectManagementApplication.java

1. com.code.employee.audit

* AuditorAwareImpl.java

1. com.code.employee.config

* DataSourceConfig.java
* JpaConfig.java
* SecondaryJpaConfig.java

1. com.code.employee.controllers

* DepartmentController.java
* EmployeeController.java

1. com.code.employee.entity

* Employee.java
* EmployeeDTO.java

1. com.code.employee.projection

* EmployeeProjection.java
* EmployeeValueProjection.java

1. com.code.employee.repositories

* EmployeeRepository.java

**Key Concepts**

1. **Spring Boot Auto-Configuration:**
   * **Auto-Configuration**: Spring Boot provides auto-configuration features that automatically configure beans relevant to the context based on the dependencies available in the classpath.
   * **Entity Management**: Two separate JPA configurations are provided to manage entities in different databases.
2. **Externalizing Configuration:**
   * **application.properties**: Configuration properties are externalized using the application.properties file, allowing easy modification without altering the codebase. The file contains configurations for both the primary and secondary data sources.
3. **DataSource Configuration:**
   * **Primary DataSource**: Manages the Employee entities.
   * **Secondary DataSource**: Manages the Department entities.
   * Each data source has its own DataSourceConfig class, JPA configuration, and transaction manager.

**Implementation Steps**

1. **Set Up Primary DataSource Configuration:**
   * **DataSourceConfig.java**: Defines the primary data source bean (primaryDataSource) and marks it as the default with the @Primary annotation. The data source properties are loaded from the application.properties file with the prefix spring.datasource.primary.
2. **Set Up Secondary DataSource Configuration:**
   * **SecondaryJpaConfig.java**: Defines the secondary data source bean (secondaryDataSource) for managing Department entities. The configuration is similar to the primary data source but uses a different prefix (spring.datasource.secondary) in the application.properties file.
3. **Configure JPA for Multiple Data Sources:**
   * **JpaConfig.java**: Handles the JPA configuration for the primary data source, including setting up the EntityManagerFactory and transaction manager.
   * **SecondaryJpaConfig.java**: Mirrors JpaConfig.java but configures it for the secondary data source.
4. **Externalize Configuration:**
   * **application.properties**: Contains properties for both data sources, ensuring that each one connects to its designated in-memory H2 database.
5. **Controller Layer:**
   * **EmployeeController.java**: Provides RESTful endpoints for managing employees using the primary data source.
   * **DepartmentController.java**: Provides RESTful endpoints for managing departments using the secondary data source.
6. **Testing Configuration:**
   * The application can be tested using the endpoints provided in the controllers. The H2 console can be used to inspect the contents of both databases.

**Conclusion**

This exercise demonstrates how to customize Spring Boot's auto-configuration to manage multiple data sources within a single application. By externalizing configurations and defining separate JPA settings for each data source, the application is made flexible, modular, and easy to maintain. This setup allows the Employee and Department entities to be managed independently.