**Exercise 10: Employee Management System – Hibernate-Specific Features**

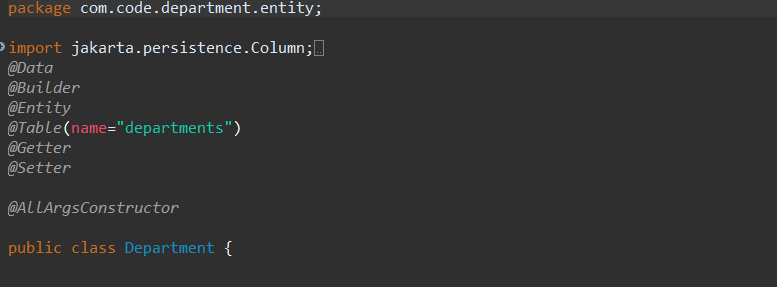
Business Scenario:

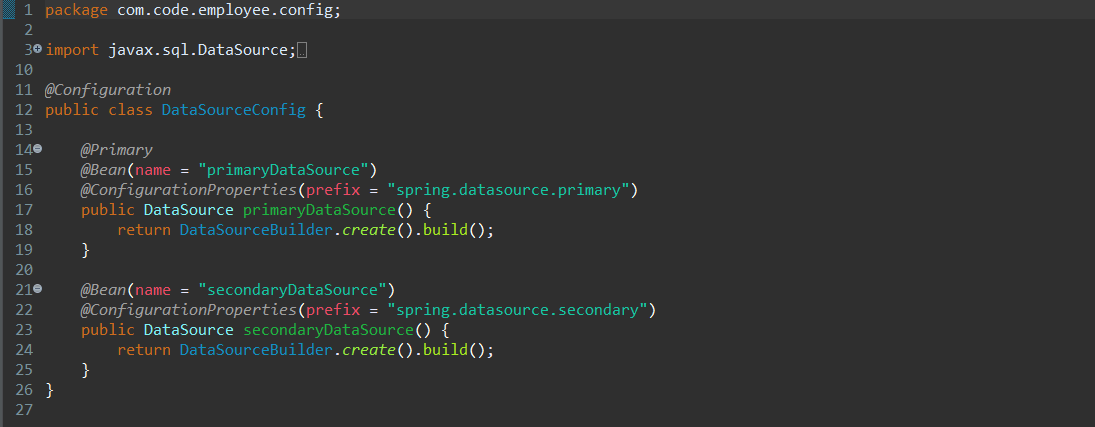
Leverage Hibernate-specific features to enhance your application's performance and capabilities.

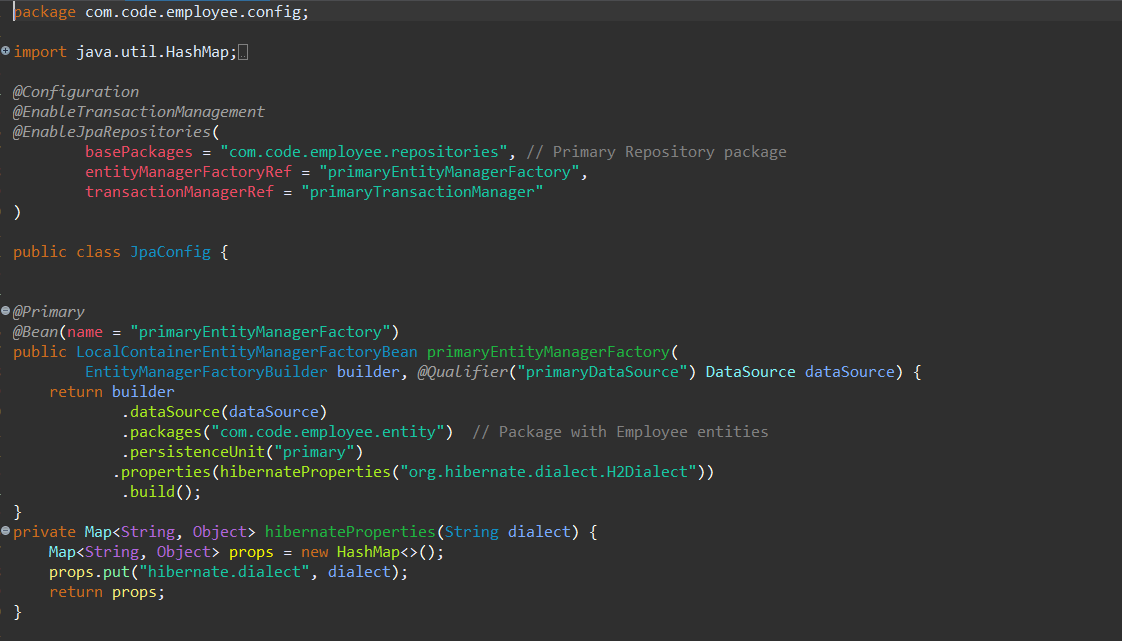
**1. Hibernate Specific Annotations:**

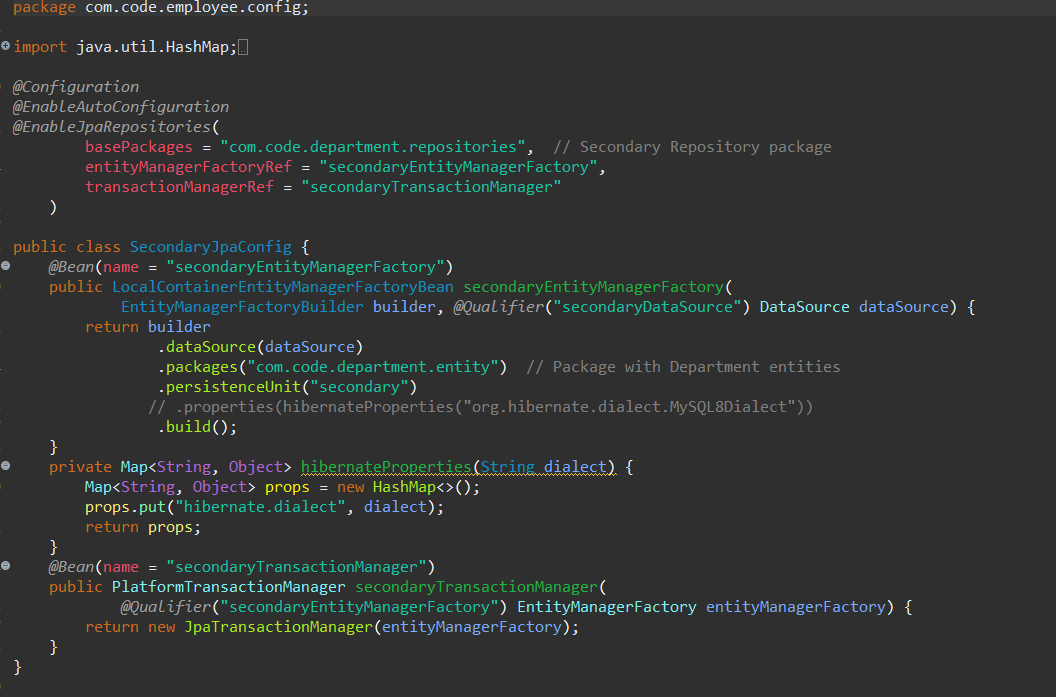
Hibernate offers several annotations that provide additional customization for entity mappings beyond standard JPA annotations. I used hibernate annotations like:









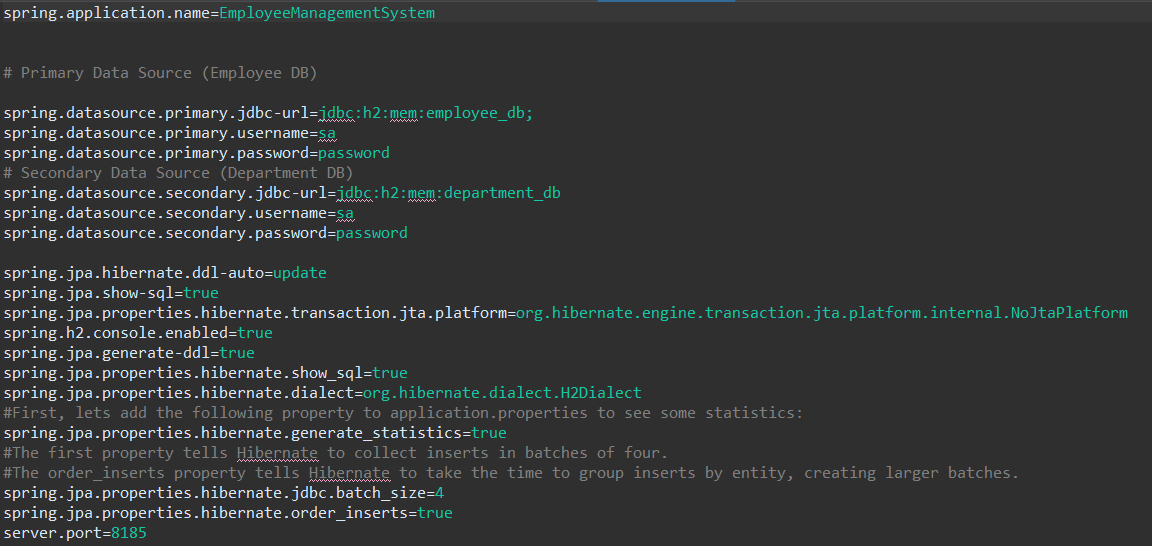
****

**2. Configuring Hibernate Dialect and Properties**

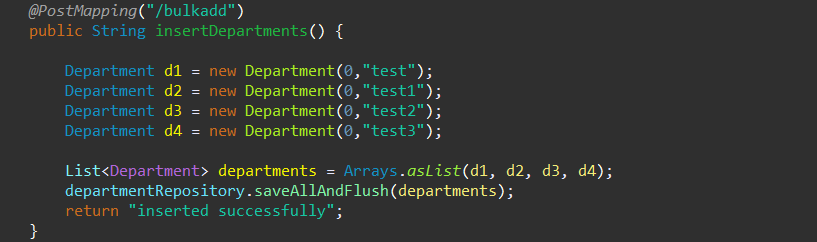
**3. Batch Processing:**

* Proper configuration of Hibernate dialect and properties can significantly improve performance and ensure compatibility with the database.
* Batch processing in Hibernate allows us to efficiently perform bulk operations by grouping multiple statements into a single batch.

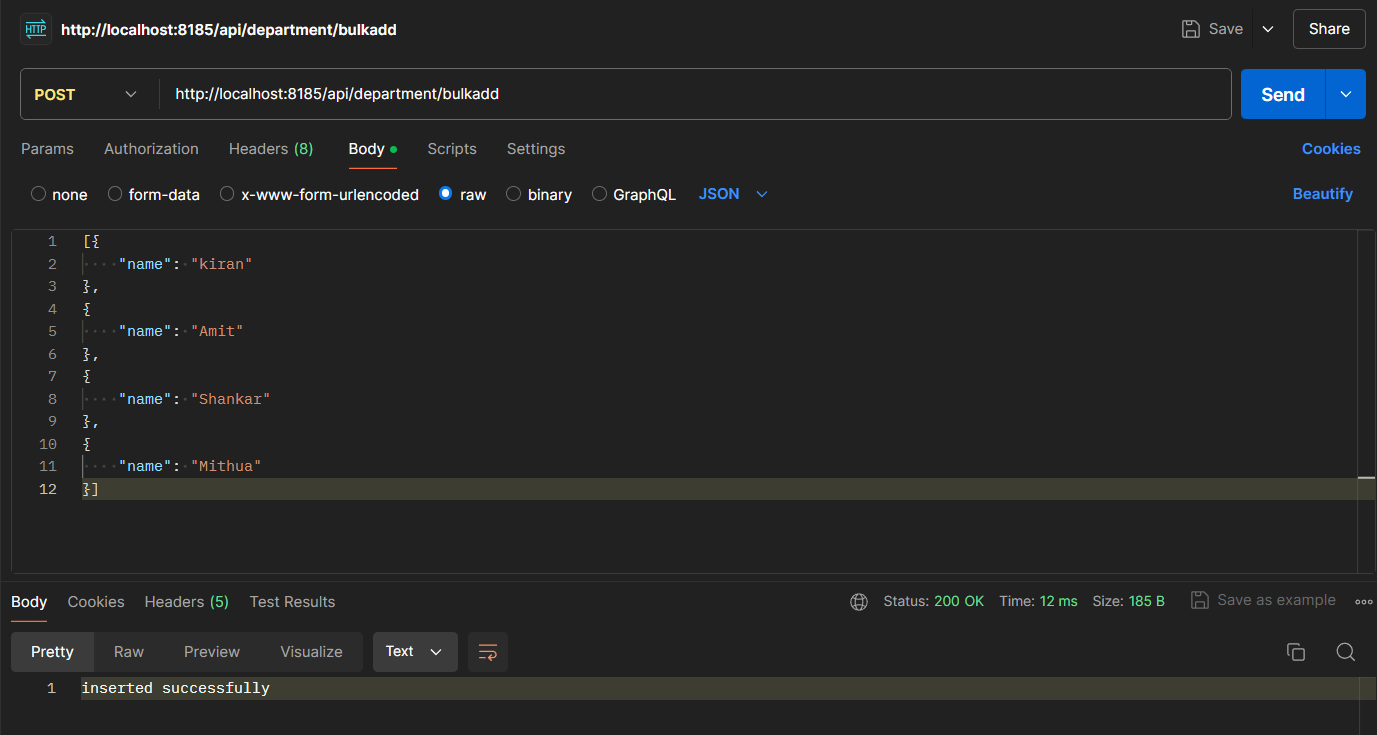
**application.Properties**

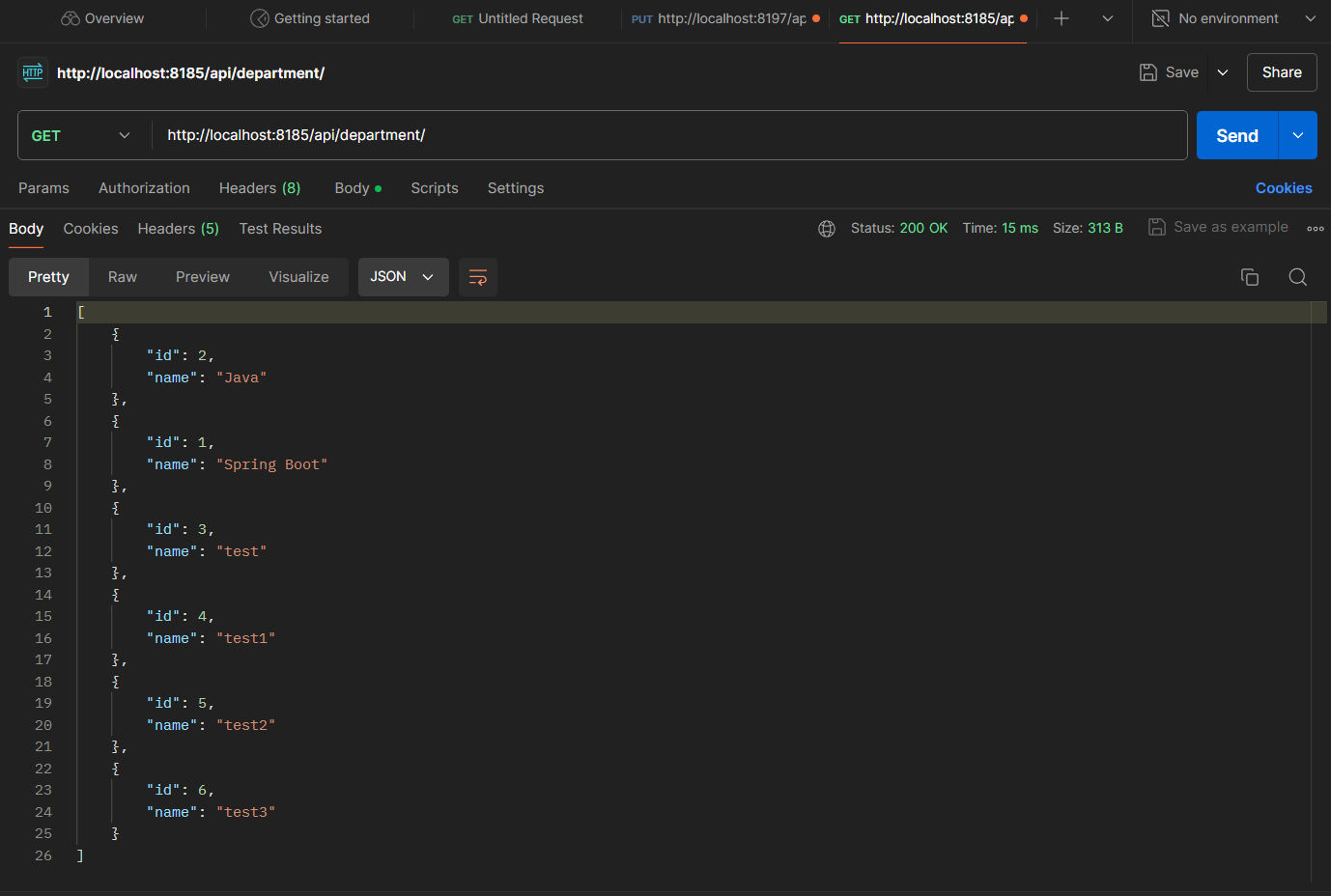
****

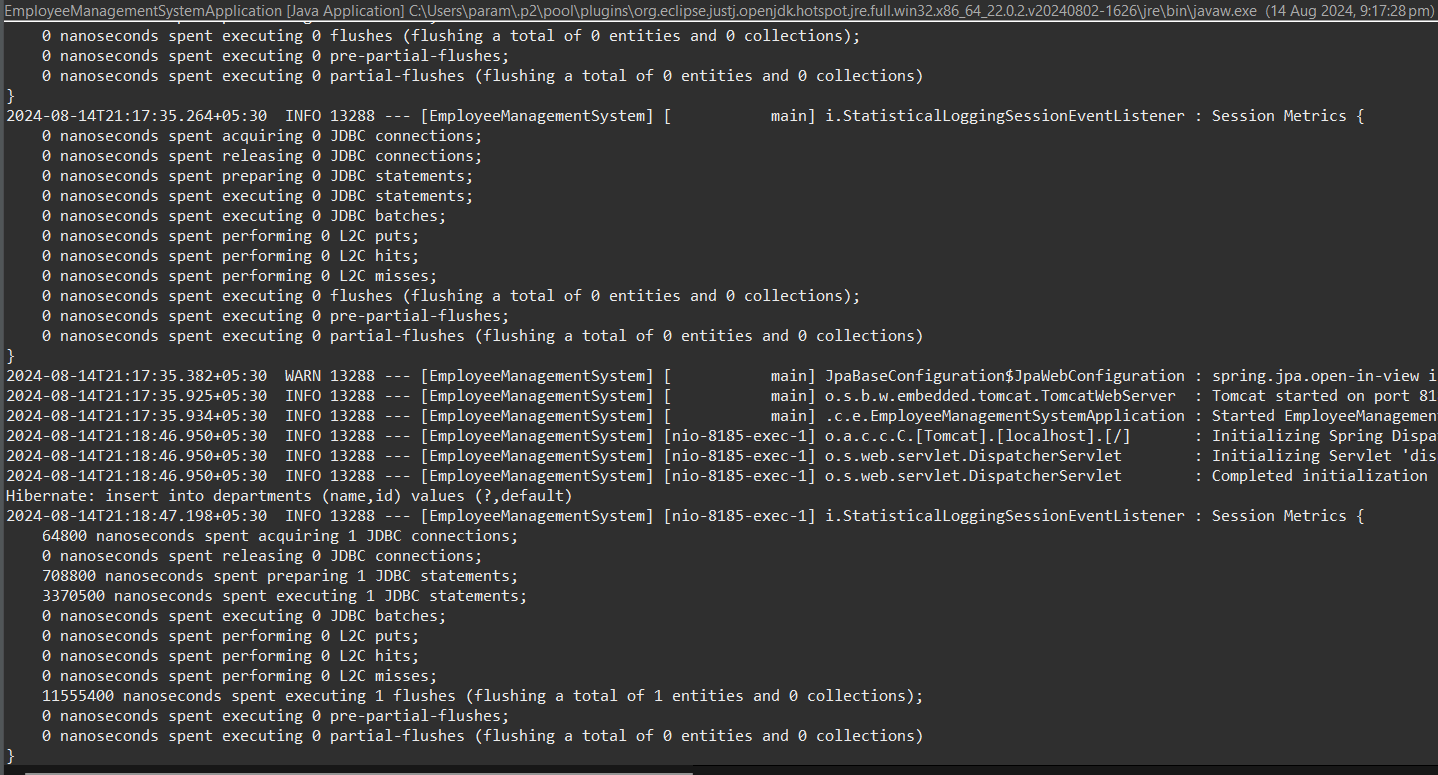
I declared a Post Mapping named “/bulkadd” for implementing Batch processing in DepartmentController.java



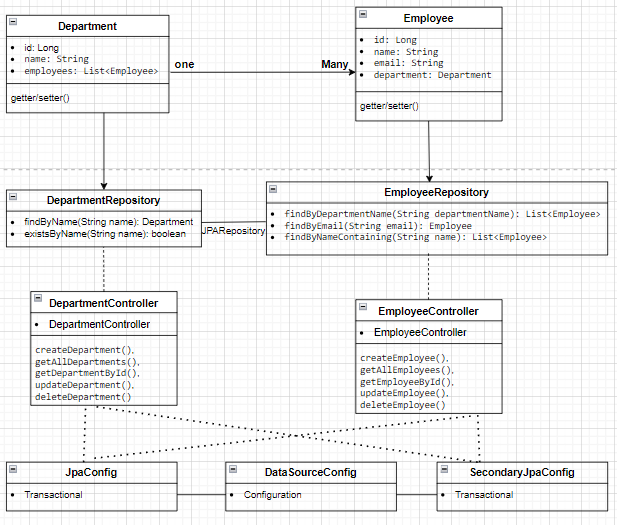
**Output:**

****

****

****

**Class Diagram:**

****

** Department Class:**

* **Attributes: Represents the id, name, and a list of Employee objects.**
* **Relationship: Has a one-to-many relationship with Employee.**

** Employee Class:**

* **Attributes: Represents the id, name, email, and a reference to the Department object.**
* **Relationship: Each Employee belongs to a single Department.**

** EmployeeRepository Interface:**

* **Methods:**
  + **findByDepartmentName(String departmentName): List<Employee>**
  + **findByEmail(String email): Employee**
  + **findByNameContaining(String name): List<Employee>**

** DepartmentRepository Interface:**

* **Methods:**
  + **findByName(String name): Department**
  + **existsByName(String name): Boolean**

** EmployeeController Class:**

* **Methods:**
* **createEmployee()**
* **getAllEmployees()**
* **getEmployeeById()**
* **updateEmployee()**
* **deleteEmployee().**

** DepartmentController Class:**

* **Methods**
* **createDepartment()**
* **getAllDepartments()**
* **getDepartmentById()**
* **updateDepartment()**
* **deleteDepartment()**

** DataSourceConfig Class:**

** JpaConfig Class:**

** SecondaryJpaConfig Class:**

**Analysis:**

** Hibernate-Specific Annotations:**

* Use @Type, @Formula, @DynamicUpdate, and @Immutable to customize entity mappings and behaviour.

** Configuring Hibernate Dialect and Properties:**

* Set the Hibernate dialect for the database.
* Adjust properties like batch\_size, order\_inserts, and order\_updates to optimize performance.

** Batch Processing:**

* Enable batch processing to handle bulk operations efficiently.
* Use the Session API to implement batch processing in the application code.

**Relationship:**

* A Department can have many Employees (OneToMany relationship).
* An Employee belongs to one Department (ManyToOne relationship).
* Department to DepartmentRepository
* Employee to EmployeeRepository
* DataSourceConfig to JpaConfig and SecondaryJpaConfiig