Employee

@Entity

@Table(name = "employees")

@DynamicUpdate

@DynamicInsert

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

// getters and setters

}

Department

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

@BatchSize(size = 10)

private List<Employee> employees = new ArrayList<>();

// getters and setters

}

application.properties

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.properties.hibernate.show\_sql=true

spring.jpa.properties.hibernate.format\_sql=true

Configuring Additional Hibernate Properties

spring.jpa.properties.hibernate.cache.use\_second\_level\_cache=true

spring.jpa.properties.hibernate.cache.use\_query\_cache=true

spring.jpa.properties.hibernate.cache.region.factory\_class=org.hibernate.cache.jcache.JCacheRegionFactory

spring.jpa.properties.hibernate.cache.provider\_class=org.ehcache.jsr107.EhcacheCachingProvider

spring.jpa.properties.hibernate.cache.region\_prefix=hibernate.cache

EmployeeService

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void batchInsertEmployees(List<Employee> employees) {

int batchSize = 20;

for (int i = 0; i < employees.size(); i++) {

entityManager.persist(employees.get(i));

if (i % batchSize == 0 && i > 0) {

entityManager.flush();

entityManager.clear();

}

}

}

}

application.properties

spring.jpa.properties.hibernate.jdbc.batch\_size=20

spring.jpa.properties.hibernate.order\_inserts=true

spring.jpa.properties.hibernate.order\_updates=true