@NamedQuery in Spring JPA – 2024

**Difference between @Query and @NamedQuery**

* There are two types of queries in JPA. One is dynamic and another one is static. Named query falls under the static category.
* Named queries are a way of organizing your static queries in a manner that is more readable, maintainable.
* The named queries are defined in the single place at entity class itself with each query has its unique name. That is the reason it is known as named query.
* @NamedQuery annotation can be applied only at the class level.
* Note that named queries have the global scope (can be accessed throughout the persistence unit), so every query should be uniquely identified even if you define queries for different entities.
* @NamedQuery annotation is a predefined query that we create and associate with a container-managed entity.

Example is given below

Entity Layer

@Entity(name = "Employee")

@Table(name = "Employee")

@Getter

@Setter

@ToString

@NoArgsConstructor

@NamedQuery(name = "Employee.query1", query = "SELECT e FROM Employee e ")

@NamedQuery(name = "Employee.query2", query = "SELECT e FROM Employee e order by e.city desc")

@NamedQuery(name = "User.findByName", query = "SELECT e FROM Employee e WHERE e.name = :name")

**public** **class** Employee {

@Id

@GeneratedValue

**private** Long id;

@Column(name = "name")

**private** String name;

@Column(name = "city")

**private** String city;

@Column(name = "project")

**private** String project;

@Column(name = "salary")

**private** **int** salary;

@Column(name = "age")

**private** **int** age;

**public** Employee(String name, String city, String project, **int** salary, **int** age) {

**super**();

**this**.name = name;

**this**.city = city;

**this**.project = project;

**this**.salary = salary;

**this**.age = age;

}

}

**Repository Layer**

@Repository

**public** **interface** EmployeeRepository **extends** CrudRepository<Employee, Long> {

@Query(name = "Employee.query1") // Name declared in @NamedQuery

List<Employee> getAll();

@Query(name = "Employee.query2") // Name declared in @NamedQuery

List<Employee> getAllInDescendingOrderOfCity();

// findByName has been declared as @NamedQuery(name = "User.findByName", ...)

List<Employee> findByName(@Param("name") String name);

// Normal query without the use of NamedQuery

@Query("SELECT e FROM Employee e WHERE e.name in :names")

List<Employee> findByNames(@Param("names") List<String> names);

}

**Service Layer**

**public** **interface** EmployeeService {

**void** saveAllEmployees(List<? **extends** Employee> empList);

List<Employee> getAllEmployee();

List<Employee> getAllEmployeeByDescendingOrderOfCity();

List<Employee> getByName(String name);

List<Employee> getEmployeesByTheirNames(List<String> names);

}

**Service Implementation Class**

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

@Autowired

**private** EmployeeRepository empRepo;

**public** **void** saveAllEmployees(List<? **extends** Employee> empList) {

empRepo.saveAll(empList);

}

@Override

**public** List<Employee> getAllEmployee() {

**return** empRepo.getAll();

}

@Override

**public** List<Employee> getAllEmployeeByDescendingOrderOfCity() {

**return** empRepo.getAllInDescendingOrderOfCity();

}

@Override

**public** List<Employee> getByName(String name) {

**return** empRepo.findByName(name);

}

**public** List<Employee> getEmployeesByTheirNames(List<String> names) {

**return** empRepo.findByNames(names);

}

}

**Autorun Class**

@Component

**public** **class** AutoRun {

@Autowired

**private** EmployeeService empService;

**public** **void** saveEmployeeDetails() {

List<Employee> empList = List.*of*(**new** Employee("John", "Bangalore", "Radina", 2000, 23),

**new** Employee("Vidya", "Chennai", "BAT", 3000, 27),

**new** Employee("Ramesh", "Kottayam", "Appllo Mission", 5000, 29),

**new** Employee("Nina", "Hyderabad", "Mars Step", 2000, 33));

empService.saveAllEmployees(empList);

}

**public** **void** showEmployee() {

empService.getAllEmployee().forEach( emp-> {

System.***out***.println("Emp: "+emp);

});

System.***out***.println("Employee order by City Descending Order");

empService.getAllEmployeeByDescendingOrderOfCity().forEach( emp-> {

System.***out***.println("Emp in Descending order of city: "+emp);

});

//Find by Name

empService.getByName("Vidya").forEach( emp-> {

System.***out***.println("Emp By Name: "+emp);

});

// Get all employees by their names

List<String> names = List.*of*("Vidya","John","Nina");

empService.getEmployeesByTheirNames(names).forEach( emp-> {

System.***out***.println("Emp By Names: "+emp);

});

}

@EventListener(ApplicationReadyEvent.**class**)

**public** **void** run() {

System.***out***.println("Application running ...");

// saveEmployeeDetails();

showEmployee();

System.***out***.println("DB Operations completed ...");

}

}