



ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD

Version 2.1 ©2022





LESSON 11 SPRING DATA JPA

Project Dependencies

- Starter projects
 - Web > Spring Web
 - SQL > Spring Data JPA
 - Developer Tools > Spring Boot DevTools
- H2
- MySQL

Entity

```
@Entity
public class Student {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String name;
    private float gpa;
    public Student() {
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getGpa() {
        return gpa;
    }
    public void setGpa(float gpa) {
        this.gpa = gpa;
    }
    @Override
    public String toString() {
        return "Student{" +
            "id=" + id +
            ", name=" + name + "\" +
            ", gpa=" + gpa +
            "}";
    }
}
```

Repository

- @Repository
- @Transactional
- public class StudentPersistence {
- @PersistenceContext
- private EntityManager em;
- public long insert(Student student) {
- em.persist(student);
- return student.getId();
- }
- }

Command Line Runner

- @Component
- public class StudentCommadLineRunner implements CommandLineRunner {
- @Autowired
- private StudentPersistence studentPersistence;
- @Override
- public void run(String... args) throws Exception {
- studentPersistence.insert(new Student("Jack", 3.3f));
- studentPersistence.insert(new Student("John", 2.0f));
- studentPersistence.insert(new Student("Jill", 3.5f));
- studentPersistence.insert(new Student("Jim", 3.0f));
- }
- }

Spring Boot H2 Configuration

- `spring.datasource.url=jdbc:h2:mem:testdb`
- `spring.datasource.driverClassName=org.h2.Driver`
- `spring.datasource.username=sa`
- `spring.datasource.password=password`



SPRING DATA

Create Interfaces

Repository

- public interface StudentRepository extends JpaRepository<Student, Long> {
- }

Command Line Runner

```
@Component
public class StudentCommnadLineRunner
implements CommandLineRunner {
    @Autowired
    private StudentRepository studentRepository;
    @Override
    public void run(String... args) throws
Exception {
        studentRepository.save(new
Student("Jack", 3.3f));
        studentRepository.save(new
Student("John", 2.0f));
        studentRepository.save(new
Student("Jill", 3.5f));
        studentRepository.save(new
Student("Jim", 3.0f));
    }
}
```

How is the Repository Working

- Is there still em? emf?
- Is the interface being implemented?
- Who is implementing the interface?
- Spring boot? Spring core? Spring Data?
- What can Spring Data do?
 - CRUD
 - Queries?
- What if I need to do something Spring data is not doing (complex query)?

- JpaRepository<Entity, Id>

- Save(student)
- FindAll()
- FindById(1l)
- Add to repository
 - `public List<Student> findByName(String name);`

application.properties (Spring Boot Configuration)

H2

- `spring.datasource.url=jdbc:h2:mem:testdb`
- `spring.datasource.driverClassName=org.h2.Driver`
- `spring.datasource.username=sa`
- `spring.datasource.password=password`

MySQL

- `spring.datasource.url=jdbc:mysql://localhost:3306/cs544db`
- `spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver`
- `spring.datasource.username=ea`
- `spring.datasource.password=cs544`

Queries

- Check the documentation
- Dynamic Queries
 - Repository
 - `@Query("SELECT s FROM Student s WHERE s.gpa >= ?1")`
 - `List<Student> findStudentsThatPassed(float passingGpa);`
- Named Queries
 - Entity
 - `@NamedQuery(name= "Student.findFailing", query="SELECT s FROM Student s WHERE s.gpa < 3.0")`
 - Repository
 - `List<Student> findFailing();`
- Native Queries
 - Repository
 - `@Query("SELECT * FROM Student WHERE gpa >= ?1", nativeQuery = true)`
 - `List<Student> findStudentsThatPassedNative(float passingGpa);`

Criteria API

Specification

```
public class StudentSpecification {
    public static Specification<Student>
    hasGpaMoreThan(float gpa) {
        return new Specification<Student>() {
            @Override
            public Predicate
            toPredicate(Root<Student> root, CriteriaQuery<?>
            query, CriteriaBuilder criteriaBuilder) {
                return
                criteriaBuilder.greaterThan(root.get("gpa"), 3.0);
            }
        };
    }
    public static Specification<Student>
    hasGpaGreaterThan(float gpa) {
        return (root, query, builder) ->
        builder.greaterThan(root.get("gpa"), gpa);
    }
}
```

Repository & Usage

```
public interface StudentRepository extends
JpaRepository<Student, Long>,
JpaSpecificationExecutor<Student> {
```

■ ----

```
studentRepository.findAll(hasGpdMoreThan(3.0f));
studentRepository.findAll(hasGpaGreaterThan(3.0f
));
```

Implement CRUD Operations

- `studentRepository.findAll()`
- `studentRepository.findById(id)`
- `studentRepository.delete(student)`
- `studentRepository.deleteById(id)`
- `studentRepository.save(student)`