# CTS NURTURE 4.0\_ JAVA FSE

Skill: Spring Data JPA with Spring Boot, Hibernate

Assignment: Spring-Data-JPA\_Quick Example

File Name: spring-data-jpa-handson

## Spring Data JPA Quick Example

This document demonstrates a basic hands-on implementation of Spring Data JPA integrated with Spring Boot and Hibernate.   
The example illustrates how to perform basic CRUD operations using JPA repository abstraction with entity mapping and service layers.   
The following sections include the project structure and key source files involved in this example.

### 1. Project Structure

The Maven-based Spring Boot project includes the following components:  
  
- Model: Defines the entity class.  
- Repository: Extends JPARepository for database operations.  
- Service: Contains the business logic.  
- Controller: Handles HTTP requests.  
- Main Application: Boots the Spring application.  
- application.properties: Configures datasource and JPA properties.

### 2. Entity Class - Student.java

@Entity  
@Table(name = "students")  
public class Student {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String name;  
 private String course;  
  
 // Getters and Setters  
}

### 3. Repository Interface - StudentRepository.java

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

### 4. Service Layer - StudentService.java

@Service  
public class StudentService {  
 @Autowired  
 private StudentRepository studentRepository;  
  
 public Student saveStudent(Student student) {  
 return studentRepository.save(student);  
 }  
  
 public List<Student> getAllStudents() {  
 return studentRepository.findAll();  
 }  
}

### 5. Controller Layer - StudentController.java

@RestController  
@RequestMapping("/api/students")  
public class StudentController {  
 @Autowired  
 private StudentService studentService;  
  
 @PostMapping  
 public ResponseEntity<Student> createStudent(@RequestBody Student student) {  
 return new ResponseEntity<>(studentService.saveStudent(student), HttpStatus.CREATED);  
 }  
  
 @GetMapping  
 public List<Student> getAllStudents() {  
 return studentService.getAllStudents();  
 }  
}

### 6. Main Class - SpringDataJpaQuickApplication.java

@SpringBootApplication  
public class SpringDataJpaQuickApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(SpringDataJpaQuickApplication.class, args);  
 }  
}

### 7. application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/cts\_students  
spring.datasource.username=root  
spring.datasource.password=your\_password  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true

## 8. Unique Project Description - Course Management System

This Spring Data JPA Quick Example demonstrates a mini Course Management System, where students can register with their name and enrolled course.   
The system saves the data into a MySQL database using Spring Boot, Spring Data JPA, and Hibernate ORM.   
RESTful APIs allow you to create and retrieve student records easily.

## 9. Output Screenshot

The following is a sample output screenshot of the API running successfully in Postman after saving a new student record:



