Spring Boot Fundamentals and REST Basics

Spring Boot Introduction
- Understanding Spring vs Spring Boot
- Spring Boot advantages and features
- Setting up development environment
- Creating your first Spring Boot application
- Understanding application.properties/yaml configuration
- purpose and understanding about pom.xml .

Spring Core Concepts
- Dependency Injection
- Inversion of Control
- Spring Bean lifecycle and application context
- Component scanning
- Common annotations (@Component, @Autowired..etc)
- MVC pattern and MVC lifecycle

REST API Basics
- REST principles and best practices
- Creating REST controllers
- @RestController and @RequestMapping
- HTTP methods (@GetMapping, @PostMapping, etc.)
- Request/Response handling
- Basic exception handling

Database Integration and JPA

Spring Data JDBC

Spring Data JPA
- JPA fundamentals
- Entity mapping
- Repository pattern
- CrudRepository and JpaRepository
- Basic JPQL queries
- @Query annotation

Database Configuration
- Setting up database (H2)
- Database configuration in application.properties

Projects (Beginner to Advanced)

1. *Task Management API*
  - Description: Build a REST API for managing tasks with basic CRUD operations
  - Key concepts:
    - REST controllers
    - Basic JPA operations
    - Request/Response handling
    - Basic error handling
  - Features:
    - Create, read, update, delete tasks
    - Task status management
    - Basic validation

Daily Learning Tips

1. *Hands-on Practice*
   - Code along with tutorials
   - Experiment with different features
   - Break things and fix them
   - Use Spring Initializr for project setup

2. *Documentation*
   - Keep the official Spring documentation handy
   - Write documentation for your own code
   - Use Swagger for API documentation

3. *Testing*
   - Write tests for new features
   - Practice Test-Driven Development
   - Use debugging tools effectively

4. *Community Resources*
   - Join Spring community forums
   - Follow Spring developers on social media
   - Participate in Spring-related discussions

Remember to:
- Start small and gradually add complexity
- Commit code regularly
- Write clean, maintainable code
- Document your learning journey
- Build a portfolio of your projects