

# 14. 다음으로

#0.강의/3.스프링로드맵/6.핵심 원리 - 고급편/강의#

## 학습 내용 정리

### 전체 목차

- 1. 예제 만들기
- 2. 스레드 로컬 - ThreadLocal
- 3. 템플릿 메서드 패턴과 콜백 패턴
- 4. 프록시 패턴과 데코레이터 패턴
- 5. 동적 프록시 기술
- 6. 스프링이 지원하는 프록시
- 7. 빈 후처리기
- 8. @Aspect AOP
- 9. 스프링 AOP 개념
- 10. 스프링 AOP 구현
- 11. 스프링 AOP - 포인트컷
- 12. 스프링 AOP - 실전 예제
- 13. 스프링 AOP - 실무 주의사항
- 14. 다음으로

## 1. 예제 만들기

- 1. 예제 만들기 - 프로젝트 생성
- 1. 예제 만들기 - 예제 프로젝트 만들기 - V0
- 1. 예제 만들기 - 로그 추적기 - 요구사항 분석
- 1. 예제 만들기 - 로그 추적기 V1 - 프로토타입 개발
- 1. 예제 만들기 - 로그 추적기 V1 - 적용
- 1. 예제 만들기 - 로그 추적기 V2 - 파라미터로 동기화 개발
- 1. 예제 만들기 - 로그 추적기 V2 - 적용
- 1. 예제 만들기 - 정리

## 2. 스레드 로컬 - ThreadLocal

- 2. 스레드 로컬 - ThreadLocal - 필드 동기화 - 개발
- 2. 스레드 로컬 - ThreadLocal - 필드 동기화 - 적용
- 2. 스레드 로컬 - ThreadLocal - 필드 동기화 - 동시성 문제
- 2. 스레드 로컬 - ThreadLocal - 동시성 문제 - 예제 코드
- 2. 스레드 로컬 - ThreadLocal - ThreadLocal - 소개
- 2. 스레드 로컬 - ThreadLocal - ThreadLocal - 예제 코드
- 2. 스레드 로컬 - ThreadLocal - 스레드 로컬 동기화 - 개발
- 2. 스레드 로컬 - ThreadLocal - 스레드 로컬 동기화 - 적용
- 2. 스레드 로컬 - ThreadLocal - 스레드 로컬 - 주의사항
- 2. 스레드 로컬 - ThreadLocal - 정리

## 3. 템플릿 메서드 패턴과 콜백 패턴

- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 시작
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 예제1
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 예제2
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 예제3
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 적용1
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 적용2
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 메서드 패턴 - 정의
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 전략 패턴 - 시작
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 전략 패턴 - 예제1
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 전략 패턴 - 예제2
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 전략 패턴 - 예제3
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 콜백 패턴 - 시작
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 콜백 패턴 - 예제
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 템플릿 콜백 패턴 - 적용
- 3. 템플릿 메서드 패턴과 콜백 패턴 - 정리

## 4. 프록시 패턴과 데코레이터 패턴

- 4. 프록시 패턴과 데코레이터 패턴 - 프로젝트 생성

- 4. 프록시 패턴과 데코레이터 패턴 - 예제 프로젝트 만들기 v1
- 4. 프록시 패턴과 데코레이터 패턴 - 예제 프로젝트 만들기 v2
- 4. 프록시 패턴과 데코레이터 패턴 - 예제 프로젝트 만들기 v3
- 4. 프록시 패턴과 데코레이터 패턴 - 요구사항 추가
- 4. 프록시 패턴과 데코레이터 패턴 - 프록시, 프록시 패턴, 데코레이터 패턴 - 소개
- 4. 프록시 패턴과 데코레이터 패턴 - 프록시 패턴 - 예제 코드1
- 4. 프록시 패턴과 데코레이터 패턴 - 프록시 패턴 - 예제 코드2
- 4. 프록시 패턴과 데코레이터 패턴 - 데코레이터 패턴 - 예제 코드1
- 4. 프록시 패턴과 데코레이터 패턴 - 데코레이터 패턴 - 예제 코드2
- 4. 프록시 패턴과 데코레이터 패턴 - 데코레이터 패턴 - 예제 코드3
- 4. 프록시 패턴과 데코레이터 패턴 - 프록시 패턴과 데코레이터 패턴 정리
- 4. 프록시 패턴과 데코레이터 패턴 - 인터페이스 기반 프록시 - 적용
- 4. 프록시 패턴과 데코레이터 패턴 - 구체 클래스 기반 프록시 - 예제1
- 4. 프록시 패턴과 데코레이터 패턴 - 구체 클래스 기반 프록시 - 예제2
- 4. 프록시 패턴과 데코레이터 패턴 - 구체 클래스 기반 프록시 - 적용
- 4. 프록시 패턴과 데코레이터 패턴 - 인터페이스 기반 프록시와 클래스 기반 프록시
- 4. 프록시 패턴과 데코레이터 패턴 - 정리

## 5. 동적 프록시 기술

- 5. 동적 프록시 기술 - 리플렉션
- 5. 동적 프록시 기술 - JDK 동적 프록시 - 소개
- 5. 동적 프록시 기술 - JDK 동적 프록시 - 예제 코드
- 5. 동적 프록시 기술 - JDK 동적 프록시 - 적용1
- 5. 동적 프록시 기술 - JDK 동적 프록시 - 적용2
- 5. 동적 프록시 기술 - CGLIB - 소개
- 5. 동적 프록시 기술 - CGLIB - 예제 코드
- 5. 동적 프록시 기술 - 정리

## 6. 스프링이 지원하는 프록시

- 6. 스프링이 지원하는 프록시 - 프록시 팩토리 - 소개
- 6. 스프링이 지원하는 프록시 - 프록시 팩토리 - 예제 코드1
- 6. 스프링이 지원하는 프록시 - 프록시 팩토리 - 예제 코드2
- 6. 스프링이 지원하는 프록시 - 포인트컷, 어드바이스, 어드바이저 - 소개

- 6. 스프링이 지원하는 프록시 - 예제 코드1 - 어드바이저
- 6. 스프링이 지원하는 프록시 - 예제 코드2 - 직접 만든 포인트컷
- 6. 스프링이 지원하는 프록시 - 예제 코드3 - 스프링이 제공하는 포인트컷
- 6. 스프링이 지원하는 프록시 - 예제 코드4 - 여러 어드바이저 함께 적용
- 6. 스프링이 지원하는 프록시 - 프록시 팩토리 - 적용1
- 6. 스프링이 지원하는 프록시 - 프록시 팩토리 - 적용2
- 6. 스프링이 지원하는 프록시 - 정리

## 7. 빈 후처리기

- 7. 빈 후처리기 - 빈 후처리기 - 소개
- 7. 빈 후처리기 - 빈 후처리기 - 예제 코드1
- 7. 빈 후처리기 - 빈 후처리기 - 예제 코드2
- 7. 빈 후처리기 - 빈 후처리기 - 적용
- 7. 빈 후처리기 - 빈 후처리기 - 정리
- 7. 빈 후처리기 - 스프링이 제공하는 빈 후처리기1
- 7. 빈 후처리기 - 스프링이 제공하는 빈 후처리기2
- 7. 빈 후처리기 - 하나의 프록시, 여러 Advisor 적용
- 7. 빈 후처리기 - 정리

## 8. @Aspect AOP

- 8. @Aspect AOP - @Aspect 프록시 - 적용
- 8. @Aspect AOP - @Aspect 프록시 - 설명
- 8. @Aspect AOP - 정리

## 9. 스프링 AOP 개념

- 9. 스프링 AOP - AOP 소개 - 핵심 기능과 부가 기능
- 9. 스프링 AOP - AOP 소개 - 애스펙트
- 9. 스프링 AOP - AOP 적용 방식
- 9. 스프링 AOP - AOP 용어 정리
- 9. 스프링 AOP 개념 - 정리

## 10. 스프링 AOP 구현

- 10. 스프링 AOP 구현 - 프로젝트 생성
- 10. 스프링 AOP 구현 - 예제 프로젝트 만들기
- 10. 스프링 AOP 구현 - 스프링 AOP 구현1 - 시작
- 10. 스프링 AOP 구현 - 스프링 AOP 구현2 - 포인트컷 분리
- 10. 스프링 AOP 구현 - 스프링 AOP 구현3 - 어드바이스 추가
- 10. 스프링 AOP 구현 - 스프링 AOP 구현4 - 포인트컷 참조
- 10. 스프링 AOP 구현 - 스프링 AOP 구현5 - 어드바이스 순서
- 10. 스프링 AOP 구현 - 스프링 AOP 구현6 - 어드바이스 종류
- 10. 스프링 AOP 구현 - 정리

## 11. 스프링 AOP - 포인트컷

- 11. 스프링 AOP - 포인트컷 - 포인트컷 지시자
- 11. 스프링 AOP - 포인트컷 - 예제 만들기
- 11. 스프링 AOP - 포인트컷 - execution1
- 11. 스프링 AOP - 포인트컷 - execution2
- 11. 스프링 AOP - 포인트컷 - within
- 11. 스프링 AOP - 포인트컷 - args
- 11. 스프링 AOP - 포인트컷 - @target, @within
- 11. 스프링 AOP - 포인트컷 - @annotation, @args
- 11. 스프링 AOP - 포인트컷 - bean
- 11. 스프링 AOP - 포인트컷 - 매개변수 전달
- 11. 스프링 AOP - 포인트컷 - this, target
- 11. 스프링 AOP - 포인트컷 - 정리

## 12. 스프링 AOP - 실전 예제

- 12. 스프링 AOP - 실전 예제 - 예제 만들기
- 12. 스프링 AOP - 실전 예제 - 로그 출력 AOP
- 12. 스프링 AOP - 실전 예제 - 재시도 AOP
- 12. 스프링 AOP - 실전 예제 - 정리

## 13. 스프링 AOP - 실무 주의사항

- 13. 스프링 AOP - 실무 주의사항 - 프록시와 내부 호출 - 문제
- 13. 스프링 AOP - 실무 주의사항 - 프록시와 내부 호출 - 대안1 자기 자신 주입
- 13. 스프링 AOP - 실무 주의사항 - 프록시와 내부 호출 - 대안2 지연 조회
- 13. 스프링 AOP - 실무 주의사항 - 프록시와 내부 호출 - 대안3 구조 변경
- 13. 스프링 AOP - 실무 주의사항 - 프록시 기술과 한계 - 타입 캐스팅
- 13. 스프링 AOP - 실무 주의사항 - 프록시 기술과 한계 - 의존관계 주입
- 13. 스프링 AOP - 실무 주의사항 - 프록시 기술과 한계 - CGLIB
- 13. 스프링 AOP - 실무 주의사항 - 프록시 기술과 한계 - 스프링의 해결책
- 13. 스프링 AOP - 실무 주의사항 - 정리

## 다음으로

### 스프링 데이터(DB) 접근 기술 안내

- 데이터베이스 커넥션
  - JDBC 커넥션
  - 커넥션 풀
- 스프링 DB 트랜잭션
  - 트랜잭션 내부 원리
  - 자세한 사용법
  - 실무 주의 사항
- 다양한 데이터 접근 기술 이해
  - JDBC
  - JdbcTemplate
  - 마이바티스 - MyBatis
  - JPA
  - 스프링 데이터 JPA
  - QueryDSL
- 데이터베이스 예외 처리
  - 스프링 데이터베이스 예외 추상화
  - 올바른 예외 처리 사용법

# 로드맵 소개

## 스프링 완전 정복 시리즈(진행중)

스프링을 완전히 마스터 할 수 있는 로드맵

URL: <https://www.infllearn.com/roadmaps/373>

### 강의 목록

- 스프링 입문 - 코드로 배우는 스프링 부트, 웹 MVC, DB 접근 기술
- 스프링 핵심 원리 - 기본편
- 모든 개발자를 위한 HTTP 웹 기본 지식
- 스프링 웹 MVC 1편
- 스프링 웹 MVC 2편
- 스프링 핵심 원리 - 고급편
- 스프링 데이터(DB) 접근 기술 (예정)
- 스프링 부트 (예정)

## 스프링 부트와 JPA 실무 완전 정복 로드맵

최신 실무 기술로 웹 애플리케이션을 만들어보면서 학습하고 싶으면 [스프링 부트와 JPA 실무 완전 정복 로드맵](#)을 추천

URL: <https://www.infllearn.com/roadmaps/149>

### 강의 목록

- 자바 ORM 표준 JPA 프로그래밍
- 실전! 스프링 부트와 JPA 활용1 - 웹 애플리케이션 개발
- 실전! 스프링 부트와 JPA 활용2 - API 개발과 성능 최적화
- 실전! 스프링 데이터 JPA
- 실전! Querydsl

# 하고 싶은 이야기

## 기술적 겸손함

개발자는 기술적 겸손함이 있는 개발자와 기술적 겸손함이 없는 개발자로 나눌 수 있다.

회사 업무에 익숙해지는 단계에서 성장이 멈추는 개발자와 지속해서 성장하는 개발자의 차이는 기술적 겸손함에 있다.  
뛰어난 시니어 개발자가 되려면 기술적 겸손함이 필요하다.

### **기술적 겸손함이 없는 개발자**

내가 아는 기술로 충분히 익숙한 회사 업무를 잘 할 수 있다.

나 정도면 개발을 잘 한다 생각한다.

### **기술적 겸손함이 있는 개발자**

개발 공부는 깊이있게 하면 할 수록 더 공부할 내용이 넓고 많아진다.

실력있는 개발자들은 본인이 기술적으로 더 많이 공부해야 한다 생각한다.

시간이 지날 수록 두 개발자의 격차는 커진다.

내가 경험한 팀의 수 많은 주니어 개발자 성장 이야기