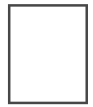


BIT BY BIT

2025/02/18 (금)

Session 14 – 디자인패턴

박설진

타임라인

10분 – 오프닝

1시간 30분 - 공유 세션 및 토의 진행

10분 – 마무리

오프닝

강의 듣는데 다들 어떠셨나요?

Ex.

다 들었는지,,

이해는 어느 정도 됐는지,,

시간이 부족하거나 남지는 않았는지 등

숙제여부 / 공부 내용 난이도 / 컨디션

공유 세션

디자인패턴 소개 ★★★	08:06
라이브러리와 프레임워크의 차이 ★★★	10:36
싱글톤 패턴 ★★★	14:44
DEEP DIVE : 싱글톤 패턴을 구현하는 7가지 방법 #1 ★★☆	12:56
DEEP DIVE : 싱글톤 패턴을 구현하는 7가지 방법 #2 ★★☆	08:36
팩토리패턴 ★★★	05:40
이터레이터패턴 ★☆☆	03:38
DI와 DIP ★★★	16:56
전략패턴 ★★★	06:58
옵저버 패턴 ★★☆	03:57
프록시 패턴 ★★★	05:36
MVC패턴과 MVP패턴 그리고 MVVM패턴 ★★★	05:22
Spring의 MVC패턴 적용사례 ★★★	03:08
flux패턴 ★★★	05:10
Q. 전략패턴과 의존성주입의 차이는 무엇인가요? ★☆☆	01:15
Q. 컨텍스트란 무엇인가요? ★☆☆	02:46

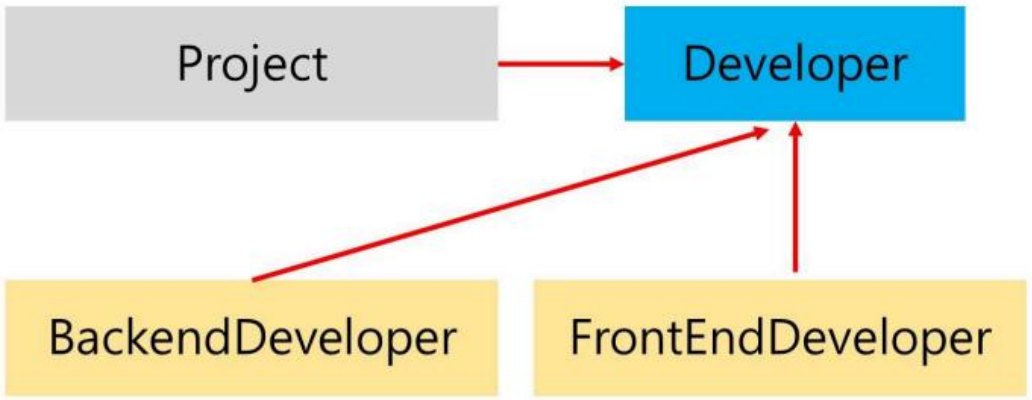
1. 라이브러리와 프레임워크

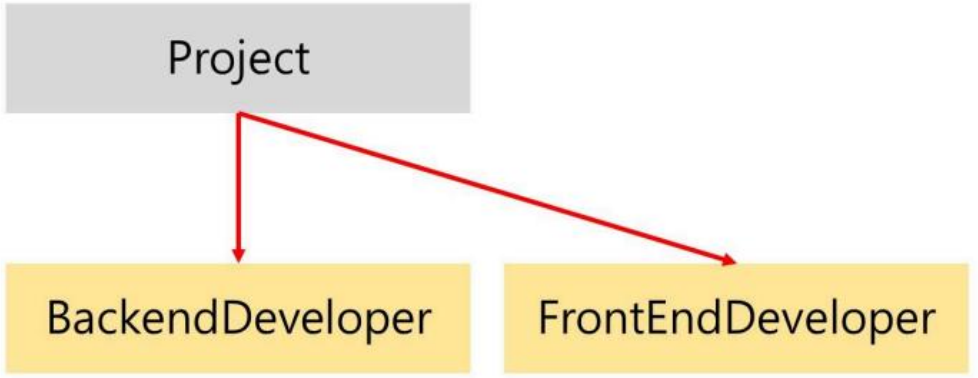
1. 라이브러리란?
2. 프레임워크란?
3. 라이브러리와 프레임워크의 차이

2. DI와 DIP

1. ‘의존한다’의 의미?
2. DI (Dependency Injection)이란?
3. DIP의 필수 규칙 두 가지
4. DI의 장과 단점

2. DI와 DIP


Project
Developer
BackendDeveloper
FrontEndDeveloper


Project
BackendDeveloper
FrontEndDeveloper

2. DI와 DIP

1. ‘의존한다’의 의미?
2. DI (Dependency Injection)이란?
3. DIP의 필수 규칙 두 가지
4. DI의 장과 단점

3. 디자인패턴

1. 디자인 패턴이란?
2. 생성패턴의 개념과 예시
3. 구조패턴의 개념과 예시
4. 행동패턴의 개념과 예시

3. 디자인패턴

1. 디자인 패턴이란?
2. 생성패턴의 개념과 예시 : 어떻게 만들지?
3. 구조패턴의 개념과 예시 : 어떻게 연결하지?
4. 행동패턴의 개념과 예시 : 어떻게 상호작용하지?

3. 디자인패턴

생성(Creational) 패턴	구조(Structural) 패턴	행위(Behavioral) 패턴
• 추상 팩토리(Abstract Factory) • 빌더(Builder) • 팩토리 메서드(Factory Methos) • 프로토타입(Prototype) • 싱글턴(Singleton)	• 어댑터(Adapter) • 브리지(Bridge) • 컴퍼지트(Composite) • 데커레이터(Decorator) • 퍼사드(Facade) • 플라이웨이트(Flyweight) • 프록시(Proxy)	• 책임 연쇄(Chain of Responsibility) • 커맨드(Command) • 인터프리터(Interpreter) • 이터레이터(Iterator) • 미디에이터(Mediator) • 메멘토(Memento) • 옵서버(Observer) • 테이트(State) • 스트래티지(Strategy) • 템플릿 메서드(Template Method) • 비지터(Visitor)

4. 싱글톤 패턴

1. 싱글톤 패턴이란?
2. 싱글톤패턴의 장점
3. 싱글톤패턴의 단점
4. Synchronized?
5. Volatile
6. enum

5. 팩토리패턴

1. 팩토리 패턴이란?
2. 팩토리 패턴의 장점

6. 이터레이터 패턴

1. 이터레이터 패턴이란?
2. 이터레이터 패턴의 장점

7. 전략 패턴

1. 전략 패턴이란?
2. 컨텍스트란?
3. 전략 패턴과 DI의 공통점
4. 전략 패턴과 DI의 차이점

8. 옵저버 패턴

1. 옵저버 패턴이란?
2. 옵저버 패턴의 실제 예시

9. 프록시 패턴

1. 프록시 패턴이란?
2. 프록시 서버를 둘 때 장점

팩토리패턴

전략패턴

싱글톤패턴

이터레이터패턴

옵저버패턴

Quiz

프록시패턴

생성(Creational) 패턴	구조(Structural) 패턴	행위(Behavioral) 패턴

Quiz

생성(Creational) 패턴	구조(Structural) 패턴	행위(Behavioral) 패턴
싱글톤패턴 팩토리패턴	프록시패턴	이터레이터패턴 전략패턴 옵저버패턴

요약

디자인 패턴	설명	장점	단점
팩토리 패턴	객체 생성 책임을 팩토리 클래스에 위임하는 패턴	객체 생성 로직 캡슐화, 유연성 제공, 확장 용이	코드 복잡도 증가, 객체 생성 방식이 숨겨져 디버깅 어려움
프록시 패턴	실제 객체에 대한 접근을 제어하는 대리 객체를 사용하는 패턴	접근 제어, 지연 로딩, 성능 향상	코드 복잡성 증가, 성능 오버헤드
이터레이터 패턴	컬렉션 객체의 원소를 순차적으로 접근할 수 있도록 하는 패턴	컬렉션과 분리된 반복 처리, 코드 간결화	성능 저하, 코드 복잡도 증가
전략 패턴	알고리즘을 클래스별로 캡슐화하여 런타임에 동적으로 선택할 수 있는 패턴	알고리즘 캡슐화, 런타임에서 알고리즘 변경 가능	클래스 수 증가, 전략 변경 시 클라이언트 코드 수정 필요
옵저버 패턴	주체 객체의 상태 변화 시 옵저버 객체에 자동으로 알림을 보내는 패턴	느슨한 결합, 자동 알림	옵저버 수가 많으면 성능 문제, 옵저버 순서 문제 발생 가능

10. MVC, MVP, MVVM

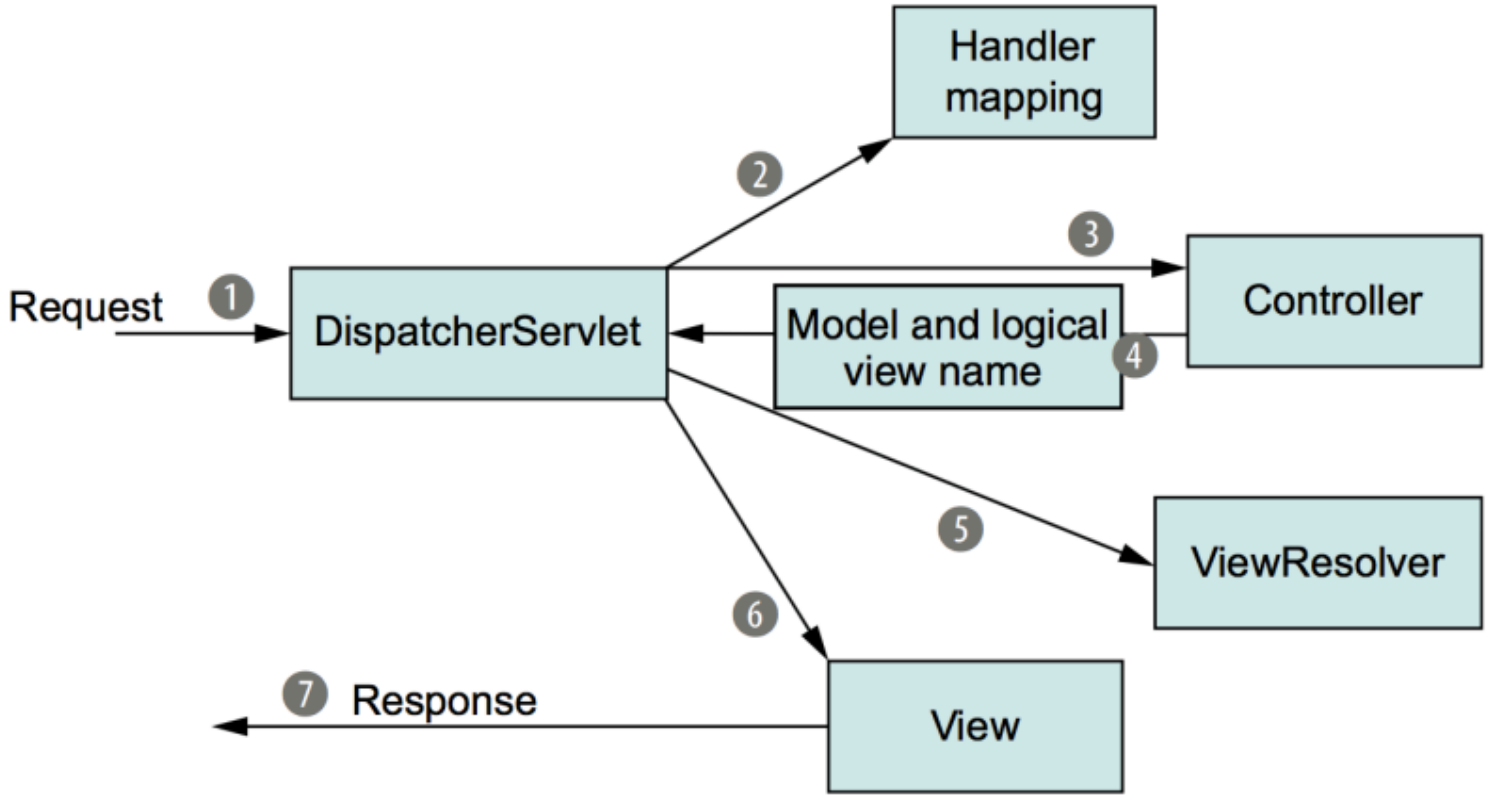
1. MVC 패턴이란?
2. MVP 패턴이란?
3. MVVM 패턴이란?

10. MVC, MVP, MVVM

차이정리

특징	MVC 패턴	MVP 패턴	MVVM 패턴
관계	컨트롤러와 뷰는 1 : n	프레젠터와 뷰는 1 : 1	뷰모델과 뷰 는 1 : n
참조	뷰는 컨트롤러를 참조 x	뷰는 프레젠터를 참조 o	뷰는 뷰모델을 참조 o

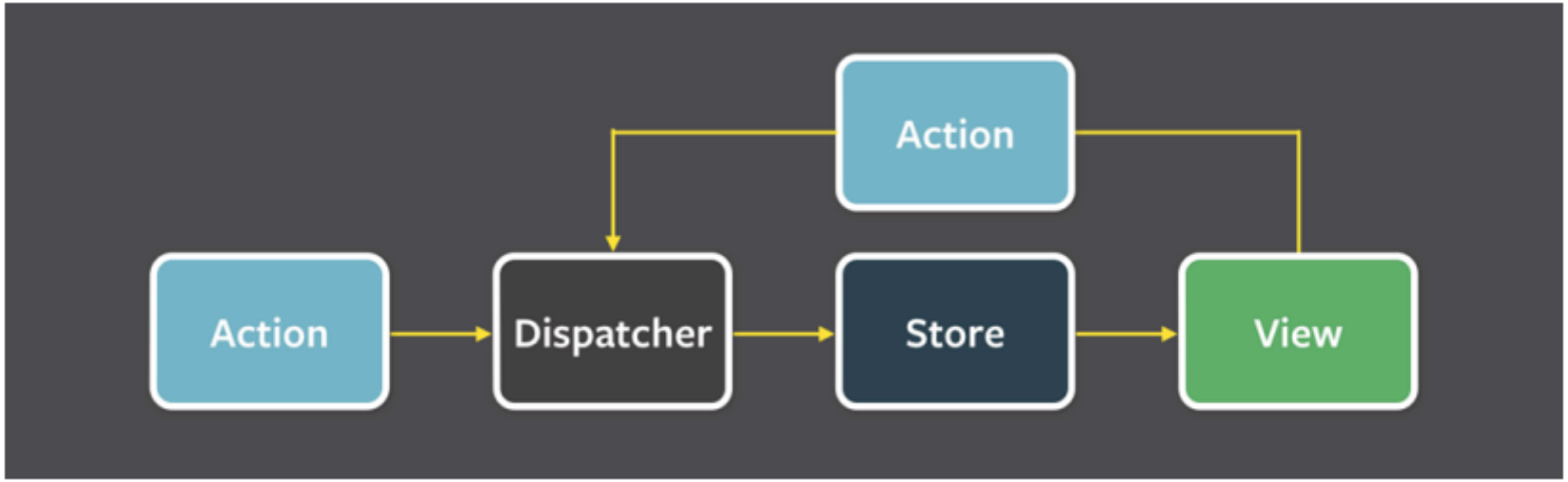
10. MVC : Spring Web MVC 처리과정



11. Flux 패턴

1. Flux 패턴이란?
2. Flux 패턴의 구성요소
3. Flux 패턴의 장점

11. Flux 패턴



감사합니다