

초격차 패키지 Online.

비즈니스 로직 구현

PART1 | @Service 이해

비즈니스 로직 구현의 시작

PART2 | Validation

입력값을 검증하는 똑똑한 방법

PART3 | 오류 처리

발생 가능한 문제 상황에 대비하기

PART4 | Spring Boot Properties

서비스 구현에 유용한 프로퍼티를 찾아보기

PART5 | 비즈니스 로직의 테스트

스프링의 테스트 지원과 목킹을 활용하기

비즈니스 로직 구현

2 Validation

Validation

2. Validation

사용자 입력값에 있을지 모르는 오류를 처리하려면?

- 모든 입력단 앞에 방어 코드가 추가됨
- 방어 코드는 복잡하고 반복적 - 대표적인 boilerplate code

```
@GetMapping("/events")
public APIDataResponse<List<EventResponse>> getEvents(
    Long placeId,
    String eventName,
    EventStatus eventStatus,
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventStartDatetime,
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventEndDatetime
) {
    if (placeId <= 0) { throw new ValidationException("placeId 는 0보다 커야 한다"); }
    if (eventName.length() < 2) { throw new ValidationException("eventName 길이는 2자 이상이어야 한다"); }

    List<EventResponse> eventResponses = eventService.getEvents(
        placeId,
        eventName,
        eventStatus,
        eventStartDatetime,
        eventEndDatetime
    ).stream().map(EventResponse::from).toList();

    return APIDataResponse.of(eventResponses);
}
```

Validation

2. Validation

애노테이션 기반으로 데이터 검증을 돕는 JSR-303, JSR-380 표준이 도입됨

- 검증 구현과 비즈니스 로직을 분리하고, 비즈니스 로직에 더 집중 가능
- 간결한 코드 표현, 더 나은 가독성

```
@GetMapping("/events")
public APIDataResponse<List<EventResponse>> getEvents(
    @Positive Long placeId,
    @Size(min = 2) String eventName,
    EventStatus eventStatus,
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventStartDatetime,
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventEndDatetime
) {
    List<EventResponse> eventResponses = eventService.getEvents(
        placeId,
        eventName,
        eventStatus,
        eventStartDatetime,
        eventEndDatetime
    ).stream().map(EventResponse::from).toList();

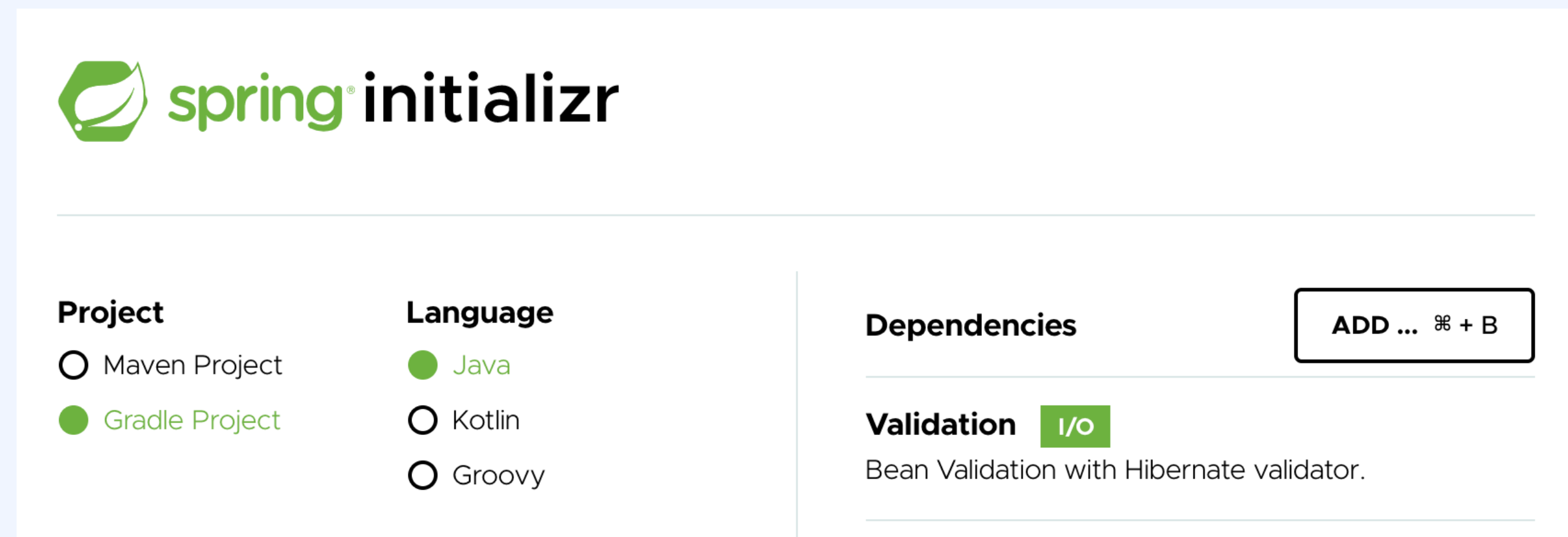
    return APIDataResponse.of(eventResponses);
}
```

Validation in Boot

2. Validation

build.gradle 에 spring-boot-starter-validation 의존성 추가

- Spring Boot 2.3 이전: spring-boot-starter-web 에 기본 포함되어 있음
- Spring Boot 2.3 이후: 직접 넣어줘야 함



The image shows the Spring Initializr web interface. At the top is the Spring logo and the text 'spring® initializr'. Below this, there are three main sections: 'Project', 'Language', and 'Dependencies'. In the 'Project' section, 'Maven Project' is selected with a radio button, and 'Gradle Project' is also visible. In the 'Language' section, 'Java' is selected with a radio button, and 'Kotlin' and 'Groovy' are also visible. In the 'Dependencies' section, there is a button that says 'ADD ... ⌘ + B'. Below this, there is a section for 'Validation' with a green 'I/O' icon and the text 'Bean Validation with Hibernate validator.'.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
```

Validation in Boot: 사용 패턴

1. @Validated + 메소드 파라미터 검증

메소드 파라미터에 validation annotation 을 직접 사용해서 검증하는 방법

- 클래스에 @Validated 필요
- 발생 예외: ConstraintViolationException, 직접 처리해 줘야 하는 예외
- ConfigurationProperties 클래스에도 적용 가능

```
@Validated
@RequiredArgsConstructor
@RequestMapping("/api")
@RestController
public class APIEventController {

    private final EventService eventService;

    @GetMapping("/events")
    public APIDataResponse<List<EventResponse>> getEvents(
        @Positive Long placeId,
        @Size(min = 2) String eventName,
        EventStatus eventStatus,
        @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventStartDatetime,
        @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime eventEndDatetime
    ) {
        List<EventResponse> eventResponses = eventService.getEvents(
```

Validation in Boot: 사용 패턴

2. Validation

2. @Valid + Data Object

검증하려는 데이터 오브젝트에만 검증 로직을 적용할 때

- @Validated 필요하지 않음
- 발생 예외: MethodArgumentNotValidException
- ResponseEntityExceptionHandler 지원을 받을 수 있음

```
@ResponseStatus(HttpStatus.CREATED)
@PostMapping("/events")
public APIDataResponse<String> createEvent(@Valid @RequestBody EventRequest eventRequest) {
    boolean result = eventService.createEvent(eventRequest.toDTO());

    return APIDataResponse.of(Boolean.toString(result));
}
```

```
public record EventRequest(
    @NotNull @Positive Long placeId,
    @NotBlank String eventName,
    @NotNull EventStatus eventStatus,
    @NotNull LocalDateTime eventStartDatetime,
    @NotNull LocalDateTime eventEndDatetime,
    @NotNull @PositiveOrZero Integer currentNumberOfPeople,
    @NotNull @Positive Integer capacity,
    String memo
) {
}
```


Reference

2. Validation

- <https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.validation>
- <https://spring.io/guides/gs/validating-form-input/>
- <https://beanvalidation.org/2.0/spec/#builtinconstraints>