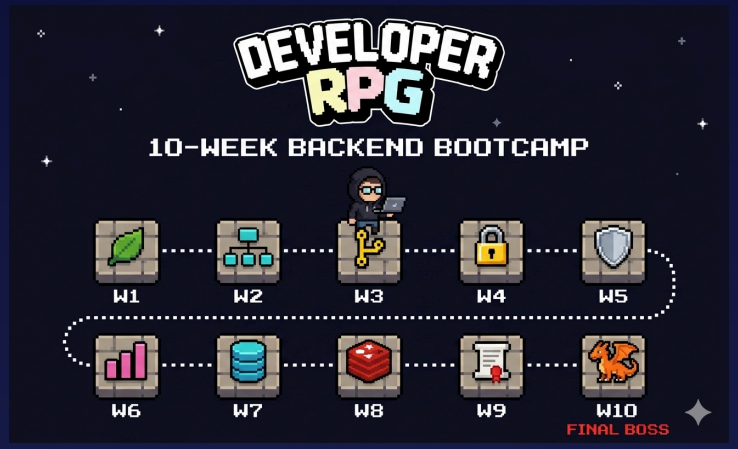

Week 1

Spring Boot 기본기

2026-06-06 (토) · 미션 공개 + 주간 방향

일단 첫 PR 💪

OT 잘 듣고 오셨나요?

체크	항목
☐	`JDK 21` 설치 확인 — `java -version`
☐	IntelliJ 프로젝트 import + Run 성공
☐	`./gradlew bootRun` 성공 로그
☐	본인 학생 레포 (`{cohort}-{username}`) clone
☐	Discord — `{cohort}-질문` / `{cohort}-리뷰` / `{cohort}-til` 참여

> *막힌 부분은 지금 손 들어주세요. 다음 슬라이드 가기 전에 같이 정리.*

⚔ QUEST 02-week1-spring-boot

- **type**: `code`
- **마감**: 2026-06-12 (금) `23:59`
- **검증**: PR → mission-guard CI green → AI 리뷰 → 점수
- **통과 조건**: PR 머지 + 팀 규칙 문서 제출

> *"Spring Boot 가 뭘 해주는지 본인 손으로 확인한다."*

이번 쭈 목표 🎯

1. **3계층 분리** — Controller / Service / Repository 책임을 본인 말로 설명
2. `@Transactional` **1개 이상** — 왜 그 위치에 붙였는지 근거
3. **4 endpoint 동작 + 테스트 3개** — 성공 / 실패 / 예외 케이스

> curriculum.md *Week 1 통과 조건과 1:1 매칭.*

핵심 개념 1 — 3계층 분리

각 계층의 책임 이 다르다.

```
Controller → Service → Repository
  HTTP         Tx          DB
  입출력      비즈니스     접근만
```

> *책임 분리를 evidence 에 본인 말로.*

핵심 개념 2 — REST 4 endpoint

게시판 (Post) — W2 로 이어진다.

메서드	경로	응답
`POST`	`/posts`	201 + body
`GET`	`/posts`	200 + 목록
`GET`	`/posts/{id}`	200 / 404
`PUT`	`/posts/{id}`	200 / 404

`@RestController` + `ResponseEntity`.

핵심 개념 3 — `@Transactional`

비즈니스 로직 경계 에 붙인다.

```
@Service
public class PostService {
  @Transactional
  public Post create(PostRequest req) {
    // 1. 검증 → 2. save → 3. 이벤트
  }
}
```

왜 Service 에? Controller 는 Tx 경계 책임 X.

> *evidence 에 이 한 줄 본인 말로.*

핵심 개념 4 — 테스트 3개

성공 / 실패 / 예외 1개씩.

```
@Test void create_success() { ... }
@Test void create_invalidTitle_400() { ... }
@Test void getById_notFound_404() { ... }
```

어노테이션	범위
`@SpringBootTest`	전체
`@WebMvcTest`	Controller
`@DataJpaTest`	Repository

이거 하면 망함 ⚠️

- ❌ Controller 안에 비즈니스 로직 → ✅ Service 위임
- ❌ `@Transactional` 을 Controller 또는 private 메서드에 → ✅ public Service 메서드에
- ❌ Repository 가 Entity 외 객체를 받음 → ✅ DTO 변환은 Service 에서
- ❌ 테스트가 success 만 → ✅ 실패 / 예외 케이스도 반드시
- ❌ Postman 캡처 없이 PR → ✅ `evidence/response-samples.md` 에 4 endpoint 응답 모두

이번 주 제출할 것 📦

```
02-week1-spring-boot/
├── report.md
├── project/                          # 실제 Spring Boot 코드
└── evidence/
    ├── api-contract.md               # 4 endpoint 명세
    ├── response-samples.md           # Postman 응답 4건
    ├── test-results.md               # 테스트 3개 결과
    ├── layer-separation-notes.md     # 3계층 분리 근거
    └── transactional-snapshot.md     # @Transactional 위치 근거
```

> *정확한 파일명은* `devcamp-submission-sample/02-week1-spring-boot/README.md`

평가는 어떻게 (5축) 

축	가중	핵심
요구사항 충족	★★	4 endpoint 모두 동작
구조	★★★	3계층 분리 근거
기술 적용	★★	`@Transactional` 위치 근거
검증 근거	★	테스트 + Postman 응답
설명력	★★	report.md + evidence

> *Week 1 은 요구사항 충족도와 설명력 에 가중. 검증 근거는 작아도 OK.*

이번 주 일정 

- **제출 마감**: 2026-06-12 (금) `23:59`
- **금 18:00**: 3주차 학생 발표자 1장 슬라이드 제출 (이번 주는 해당 없음)
- **토 15:00–16:30**: 1주차는 이 슬롯 없음 — 오프라인 저녁 OT 가 대체
- **오피스아워**: 화·목 `21:00` `{cohort}-질문` 채널 스레드
- **TIL**: 매일 한 줄씩 `{cohort}-til` — 본인 학습 로그 누적

첫 주 흐름

1. 본인 레포 clone — `git clone git@github.com:GaeChwiRpg/{cohort}-{username}.git`
2. `02-week1-spring-boot/project/` 에 Spring Boot 코드 시작 (이미 부트스트랩됨)
3. endpoint 1개 구현 → Postman 으로 즉시 확인 → 테스트 1개 → 다음 endpoint
4. 4 endpoint 완성 후 `evidence/` 5개 파일 채우기
5. `git checkout -b submit/02-week1-spring-boot` → push → PR
6. mission-guard CI green → AI 리뷰 → `{cohort}-리뷰` 채널 알림

> *막히면 뭘 시도했는지 와 함께 `{cohort}-질문` 채널.*

질문 ㄱㄱ ?

질문 ㄱㄱ. 막히면 바로 `{cohort}-질문` — 24시간 내 답.

> *다음 주: **Week 2 — JPA 연관관계 / 영속성** (14:00) + **도메인 모델링** 강의 (15:00). 1주차 코드를 그대로 이어서 리팩토링.*