잘못된 테스트 사례 학습

Java/Spring 테스트를 추가하고 싶은 개발자들의 오답노트

왜 내가 하는 TDD는 실패하는가?

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

어떻게든 돌아가는 서비스

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

성장세

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

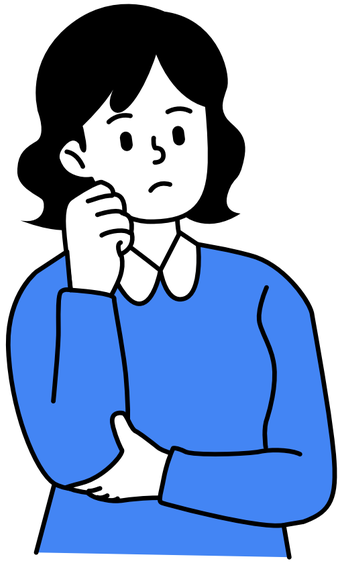
시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

개발 속도에 정체기

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?



무엇이 문제일까?

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

배포가 무섭다는 팀원

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

문제는 회귀 버그

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

테스트를 넣자

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

여기까진 좋습니다!

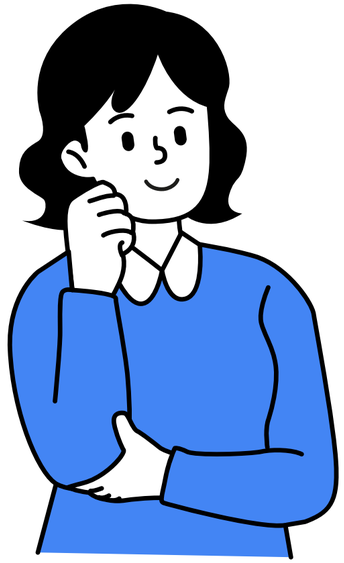
시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

가시적인 성과가 필요하다.

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?



간단한 것으로 테스트

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

만만치 않은 것의 등장...

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

그래도 일단 간다...

시작하기 앞서

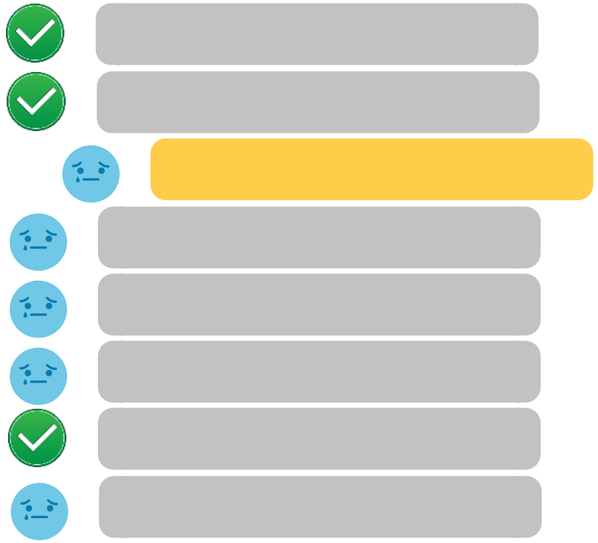
왜 내가 하는 TDD는 실패하는가?

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = {
  ServiceA.class, RepositoryA.class, RepositoryB.class, RepositoryC.class})
public class ServiceATest {

  @Autowired
  private ServiceA serviceA;
  @Autowired
  private RepositoryA repositoryA;
  @Autowired
  private RepositoryB repositoryB;
  @Autowired
  private RepositoryC repositoryC;

  @Test
  public void functionATest(){
    // given
    A a = new A();
    B b = new B();
    C c = new C();
    repositoryA.save(a);
    repositoryB.save(b);
    repositoryC.save(c);

    // when
    serviceA.functionA();

    // then
    assert(...);
  }
}
```

이게 맞아...?

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

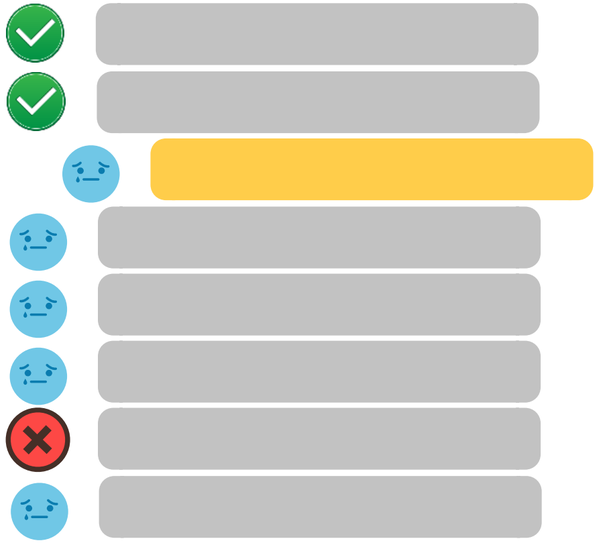


```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = {
  ServiceA.class, RepositoryA.class, RepositoryB.class, RepositoryC.class})
public class ServiceATest {

  @Autowired
  private ServiceA serviceA;
  @Autowired
  private RepositoryA repositoryA;
  @Autowired
  private RepositoryB repositoryB;
  @Autowired
  private RepositoryC repositoryC;

  @Test
  public void functionATest(){
    // given
    A a = new A();
    B b = new B();
    C c = new C();
    repositoryA.save(a);
    repositoryB.save(b);
    repositoryC.save(c);

    // when
    serviceA.functionA();

    // then
    assert(...);
  }
}
```

요령없이 짠 옛날 코드

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?



비결정적인 테스트들

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = {
  ServiceA.class, RepositoryA.class, RepositoryB.class, RepositoryC.class})
public class ServiceATest {

  @Autowired
  private ServiceA serviceA;
  @Autowired
  private RepositoryA repositoryA;
  @Autowired
  private RepositoryB repositoryB;
  @Autowired
  private RepositoryC repositoryC;

  @Test
  public void functionATest(){
    // given
    A a = new A();
    B b = new B();
    C c = new C();
    repositoryA.save(a);
    repositoryB.save(b);
    repositoryC.save(c);

    // when
    serviceA.functionA();

    // then
    assert(...);
  }
}
```

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

커버리지를 위한 테스트

시작하기 앞서

왜 내가 하는 TDD는 실패하는가?

앞선 시나리오

무엇이 문제였는가?

1. 레거시 코드에 테스트를 넣는게, TDD가 아닙니다

앞선 시나리오

무엇이 문제였는가?

2. 레거시에 테스트를 넣으려면 코드 개선이 필요합니다

테스트의 목적

1. 회귀 버그 방지
2. 유연한 설계로 개선

앞선 시나리오

무엇이 문제였는가?

2. 레거시에 테스트를 넣으려면 코드 개선이 필요합니다

테스트의 목적

1. 회귀 버그 방지
2. 유연한 설계로 개선

1. 테스트를 쉽게 만들어줌
2. 테스트를 결정적이게 만들어줌

앞선 시나리오

무엇이 문제였는가?

2. 레거시에 테스트를 넣으려면 코드 개선이 필요합니다

테스트의 목적

1. 회귀 버그 방지
2. 유연한 설계로 개선

매우 도전적이기도 함

왜냐면 우리의 시스템은 회귀 테스트가 없기 때문에

앞선 시나리오

무엇이 문제였는가?

3. 커버리지에 집착하면 안됩니다.

결과

커버리지를 올리기위한 mock 프레임워크 사용법만 고민하고,

왜 테스트를 해야하고, 어떻게 테스트 해야하는지 고민하지 않았기 때문

해설

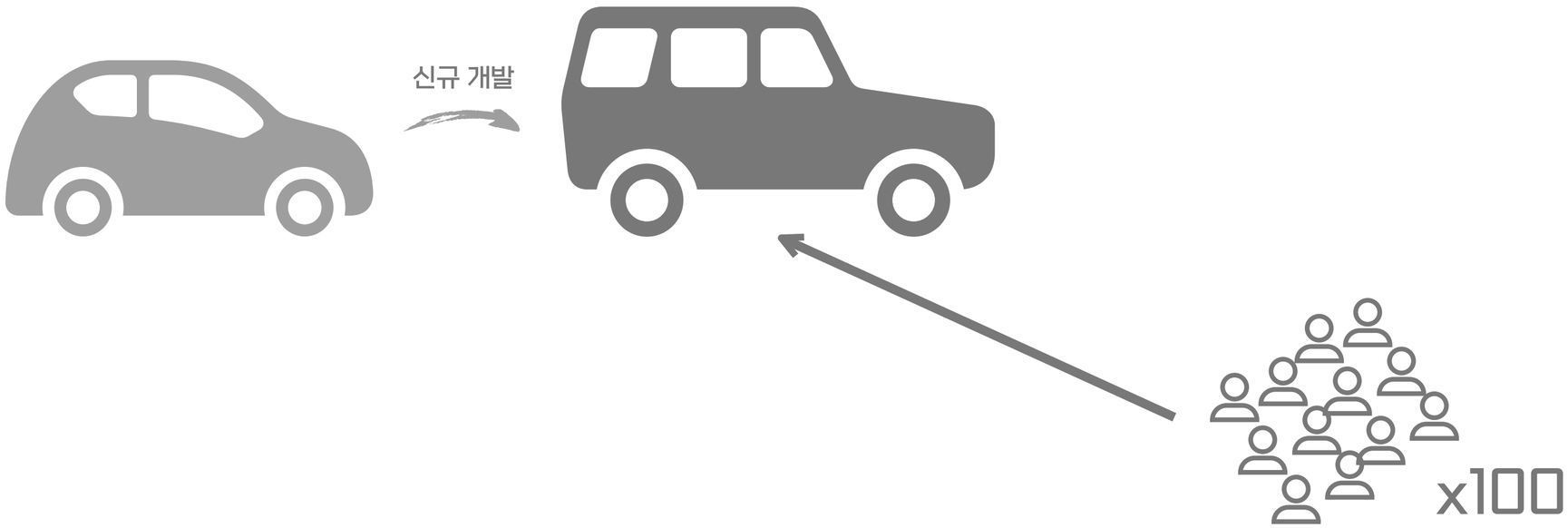
전통적인 서비스의 성장 방식

해설

전통적인 서비스의 성장 방식

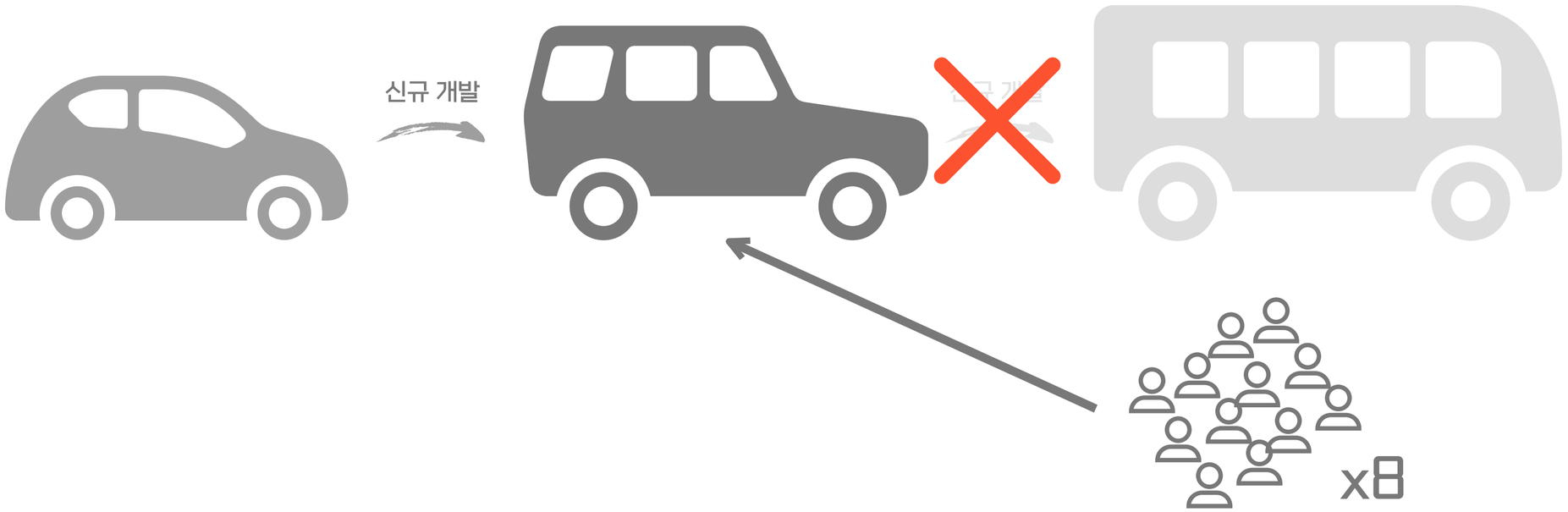
해설

전통적인 서비스의 성장 방식



해설

전통적인 서비스의 성장 방식

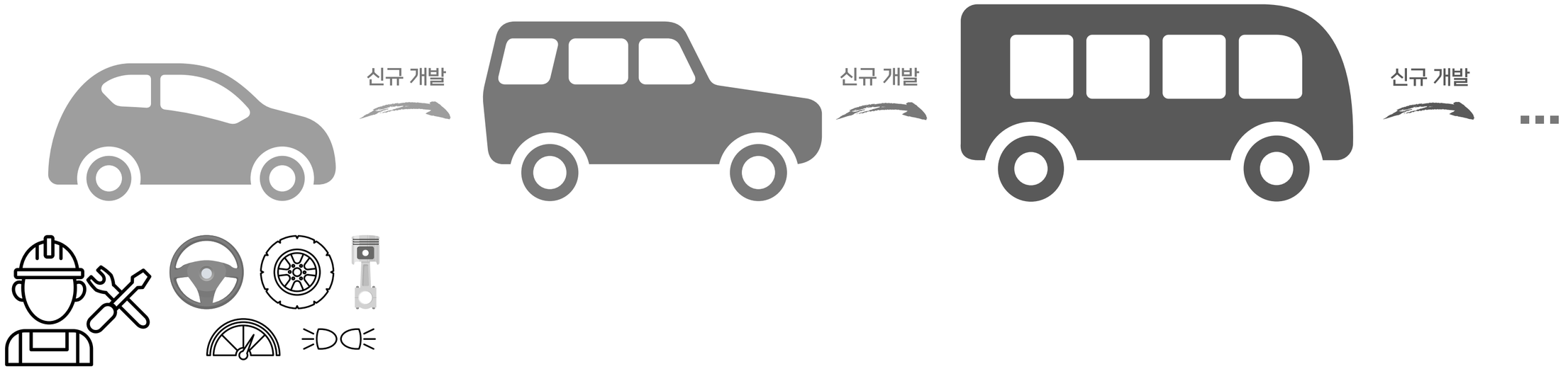


해설

전통적인 서비스의 성장 방식

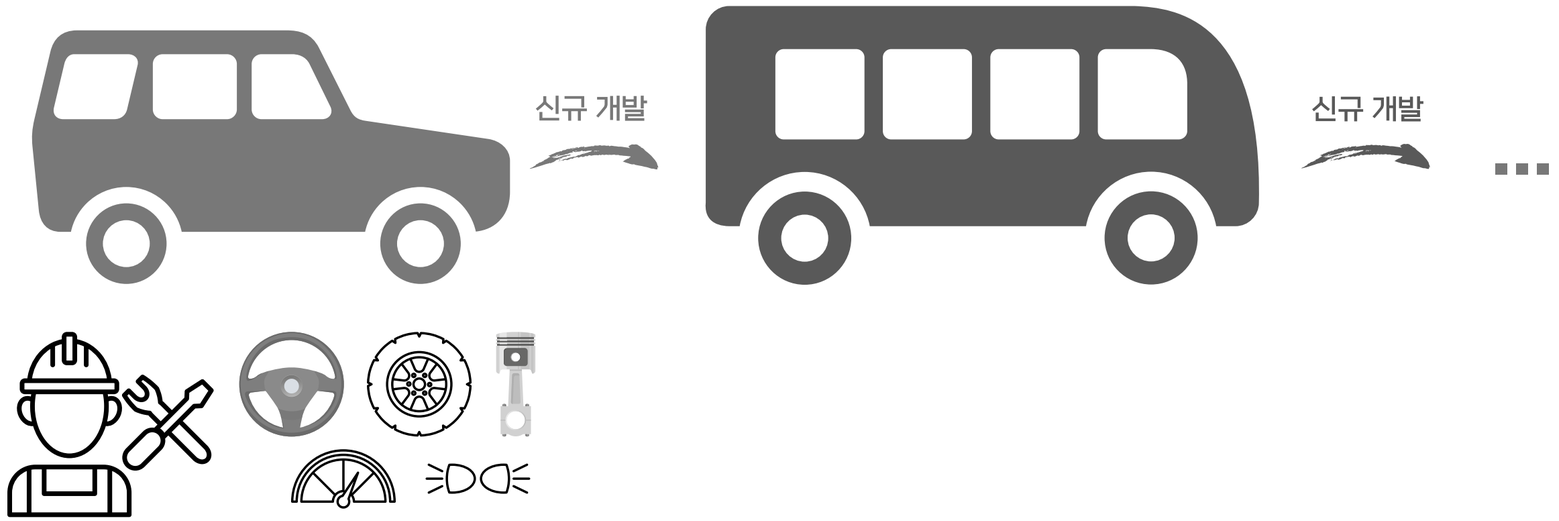
해설

전통적인 서비스의 성장 방식



해설

전통적인 서비스의 성장 방식

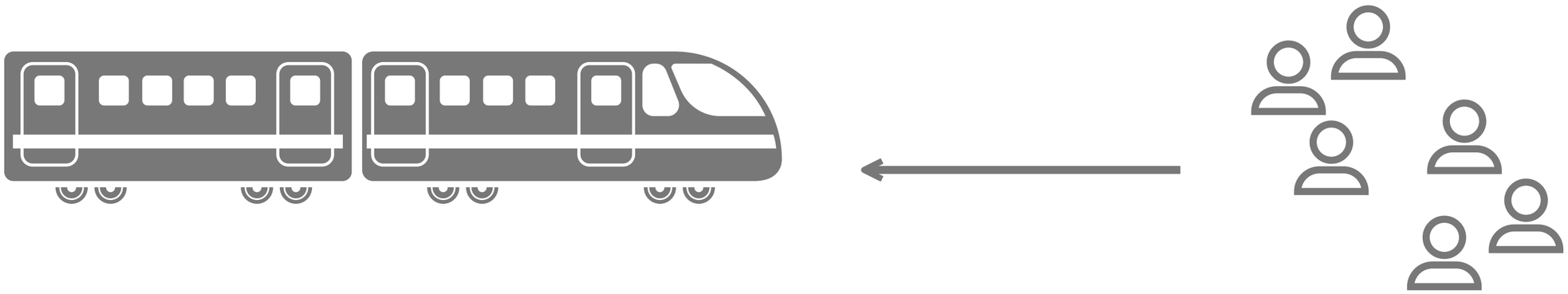


해결책

테스트와 확장성을 동시에 지닌 서비스의 성장 방식

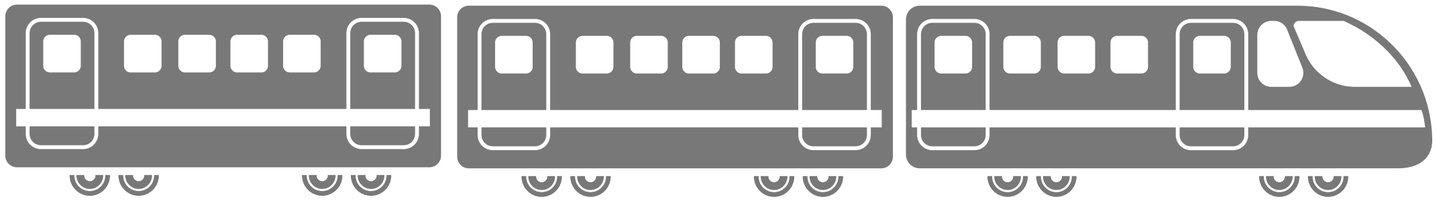
해결책

테스트와 확장성을 동시에 지닌 서비스의 성장 방식



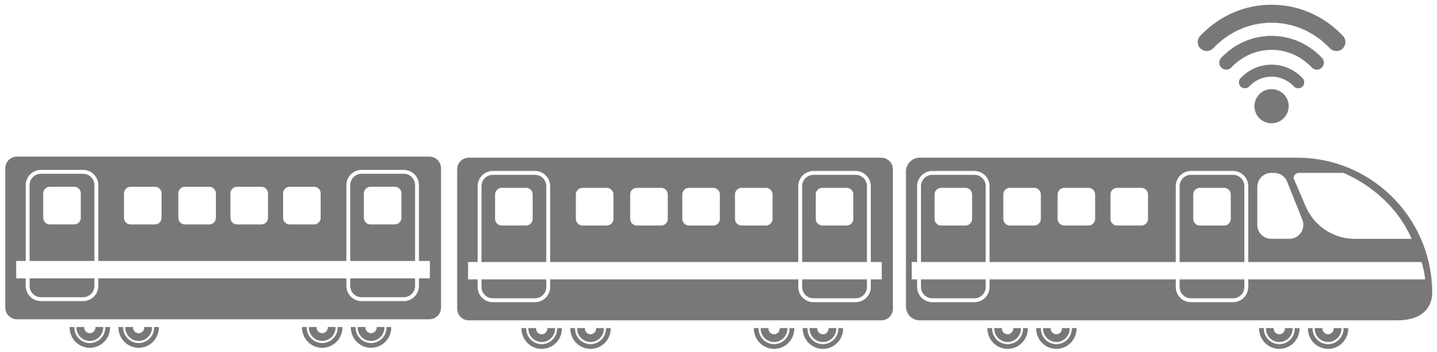
해결책

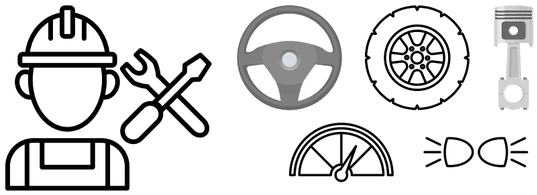
테스트와 확장성을 동시에 지닌 서비스의 성장 방식



해결책

테스트와 확장성을 동시에 지닌 서비스의 성장 방식





해결책

방향성

TDD를 논하기 전에
테스트가 가능한 구조로
변경 되어야 한다

해결책

무엇을 배울 것인가?

01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록

무엇을 배울 것인가?

01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록

무엇을 배울 것인가?

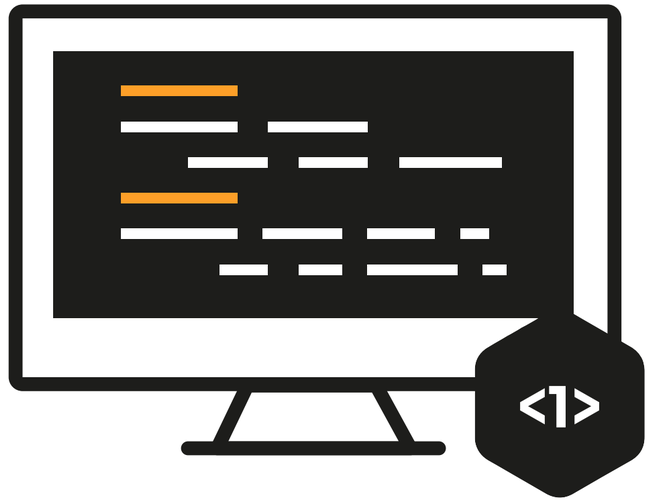
01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록



무엇을 배울 것인가?

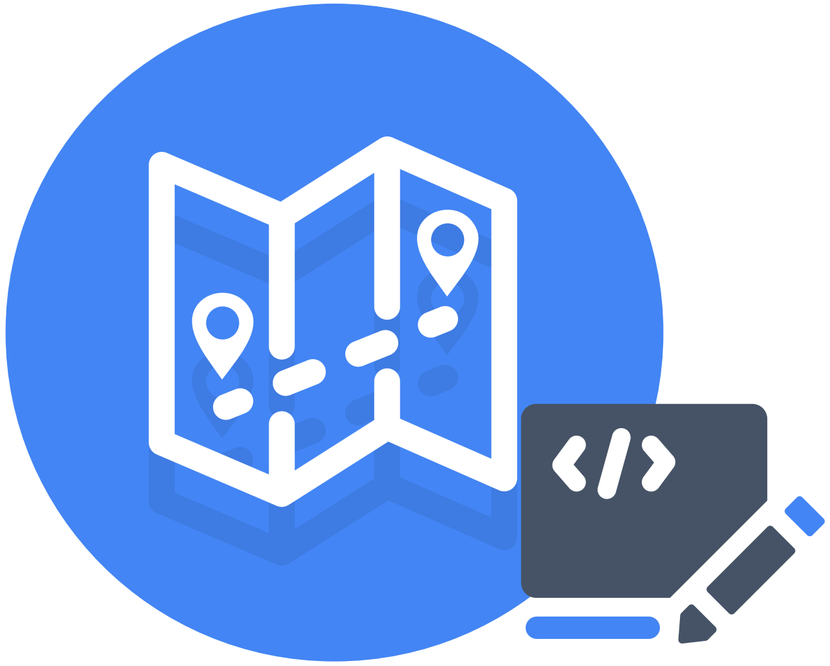
01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록



무엇을 배울 것인가?

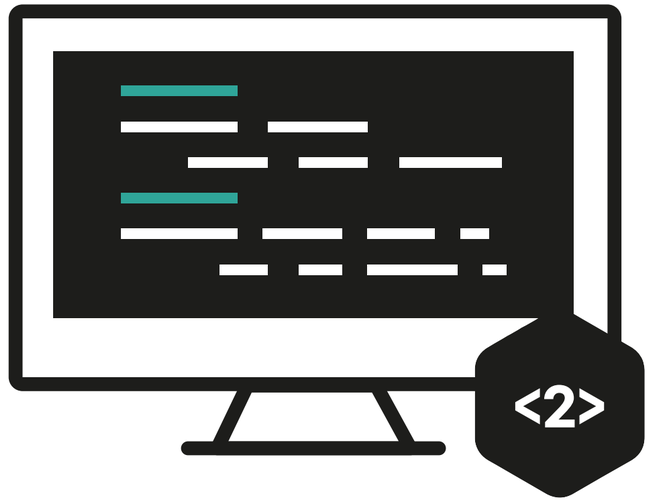
01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록



무엇을 배울 것인가?

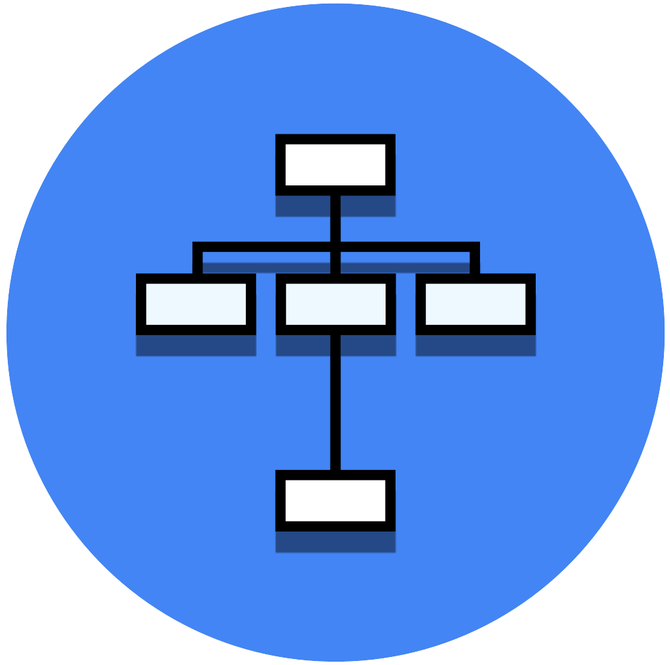
01 테스트 이론

02 실기 1부

03 방향성 탐색

04 실기 2부

05 아키텍처 & 부록



대상 독자

Spring

스프링 / 스프링 부트 사용법을 이미 알고 있는 사람

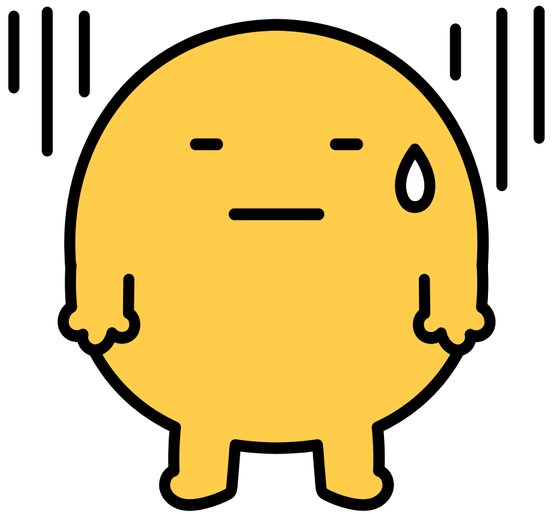
JPA

Spring data jpa를 다뤄 보신 분

약간의 테스트

테스트를 넣어보고 싶고, 테스트를 통해 설계가 어떻게 개선되는지 알고 싶으신 분

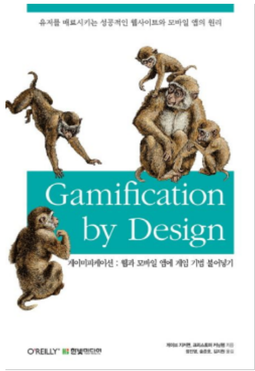
서문. TDD에 관하여



서문. TDD에 관하여

> 속된 말로 한동안 게이미피케이션에 꽂혀있었죠.

어느 순간부터 기대감은 점차 실망으로 바뀌기 시작했습니다. 게이미피케이션은 만병통치약이 아니더군요.
실제로 성공한 사례를 찾기가 힘들었습니다. 특히나 보수적인 뉴스 미디어 분야에서는 결과가 참혹했죠.
자연히 관심이 줄어들었습니다. 그러던 차에 이 책을 번역해달라는 의뢰를 받았습니다.
고민을 했습니다. "나 자신도 확신이 없는데 과연 이 책을 번역해서 널리 알리는 게 맞나?"



한계가 보임에도 불구하고 번역하기로 결정한건, 누가 뭐래도 "게임적인 사고와 기법을 활용해서 이용자를 몰입시키고 문제를 해결한다"는
게이미피케이션의 기본 생각 자체는 여전히 너무나 매력적이기 때문입니다. 게이미피케이션은 분명 보물상자가 아닙니다.
하지만 보물지도는 된다고 생각해요. 물론 지도에는 수많은 암호와 함정이 숨어있죠. 이를 현명하게 풀어나간다면 금은보화를 얻을 수 있습니다.

게이브 지커맨 , 크리스토퍼 커닝햄, 게이미피케이션: 웹과 모바일 앱에 게임 기법 불어넣기 (유저를 매료시키는 성공적인 웹사이트와 모바일 앱의 원리), 정진영, 송준호, 김지원 옮김, (한빛미디어, 2012-05-01), 7p

서문. TDD에 관하여

> 속된 말로 한동안 TDD에 꽂혀있었죠.

어느 순간부터 기대감은 점차 실망으로 바뀌기 시작했습니다. TDD는 만병통치약이 아니더군요.
실제로 성공한 사례를 찾기가 힘들었습니다.
자연히 관심이 줄어들었습니다. 그러던 차에 이런 강의를 해보는 게 어떻겠냐는 의견을 받았습니다.
고민을 했습니다. "나 자신도 확신이 없는데 과연 하는 게 맞나?"

강의를 진행하기로 결정한 건, TDD는 잘 모르겠으나 테스트의 필요성과 중요성에 대해서는 깊이 공감하고 있으며
TDD의 철학과 방법론 자체가 여전히 너무나 매력적이기 때문입니다. 제 생각에 TDD는 분명 보물상자가 아닙니다.
그리고 보물지도는 된다고 생각해요. 물론 지도에는 수많은 암호와 함정이 숨어있죠. 이를 현명하게 풀어나간다면 금은보화를 얻을 수 있습니다.

※주의사항

이전 강의 내용이 다수 중복됩니다
(특히 이론 / 방향성 파트)

※주의사항

보강 자료를 1편에 올려두었으니, 참고 부탁드립니다

※주의사항

- [ ] mockito 라이브러리 사용법을 알려드리지 않을겁니다
- [ ] TDD 강의는 아닙니다
- [ ] 종장엔 아키텍처 이야기를 합니다
- [ ] 이전 강의와 다소 중복되는 내용이 있을 수도 있습니다 (특히 이론 / 방향성 파트)

RETURN;