AutoParams를 사용한 Spring Boot 응용프로그램 테스트

이규원
와이어드컴퍼니 CTO

자료 링크

슬라이드

https://1drv.ms/p/s!ArHM66R5MeWxgsdKpHQOVTm39KXkug?e=xaPPBi

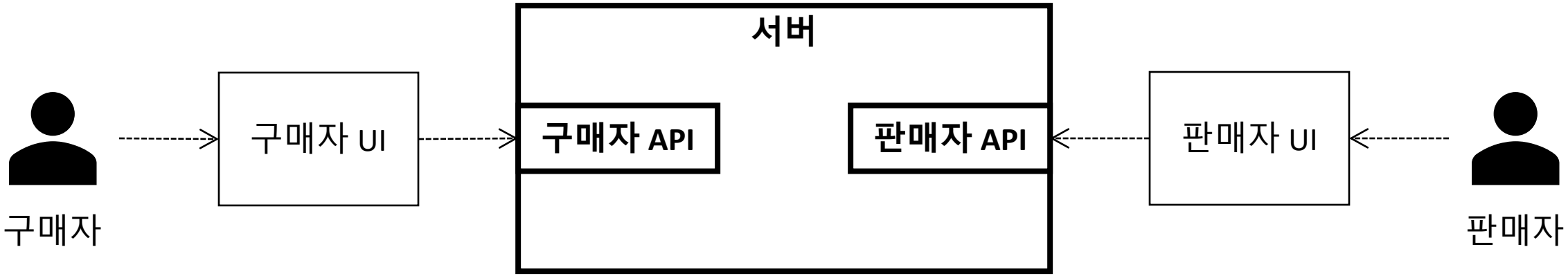
소스코드

https://github.com/gyuwon/SpringCamp2024

예제 시스템

예제 시스템 소개

- 커머스 플랫폼 서비스 서버



임의 테스트 데이터

첫번째 단위 테스트

```
@SpringBootTest(
    webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT,
    classes = CommerceApplication.class
)
@DisplayName("POST /api/consumer/signup")
public record PostTests(@Autowired TestRestTemplate client) {

    @Test
    void 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다() {
        String path = "/api/consumer/signup";
        var command = new SignUp("user@test.com", "my password");

        ResponseEntity<Void> response = client.postForEntity(path, command, Void.class);

        assertThat(response.getStatusCode().is2xxSuccessful()).isTrue();
    }
}
```

첫번째 단위 테스트 만족

```
@RestController
public class ConsumerSignUpController {

    @PostMapping("/api/consumer/signup")
    public void signUp() {
    }
}
```

첫번째 단위 테스트 결과

PostTests.올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다: 1.12 s
1 total, 1 passed

Collapse | Expand

POST /api/consumer/signup		1.12 s
올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다	passed	1.12 s

Generated by IntelliJ IDEA on 5/9/24, 3:42 PM

두번째 단위 테스트

```
@Test
void 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    String email = "user@test.com";
    String password1 = "my password 1";
    String password2 = "my password 2";
    client.postForEntity(path, new SignUp(email, password1), Void.class);

    ResponseEntity<Void> response = client.postForEntity(
        path,
        new SignUp(email, password2),
        Void.class
    );

    assertThat(response.getStatusCode().value()).isEqualTo(400);
}
```

두번째 단위 테스트 만족

```
@Entity
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Table(indexes = { @Index(columnList = "email", unique = true) })
public class ConsumerEntity {

    @Id
    @GeneratedValue
    private Long id;

    @Setter
    private String email;
}
```

두번째 단위 테스트 만족

```
@RestController
public record ConsumerSignUpController(ConsumerJpaRepository repository) {

    @PostMapping("/api/consumer/signup")
    public ResponseEntity<Void> signUp(@RequestBody SignUp command) {
        ConsumerEntity consumer = ConsumerEntity.builder().email(command.email()).build();
        try {
            repository.save(consumer);
        } catch (Exception exception) {
            return ResponseEntity.badRequest().build();
        }
        return ResponseEntity.noContent().build();
    }
}
```

두번째 단위 테스트 결과

PostTests.존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다: 1 total, 1 passed 1.22 s

Collapse | Expand

POST /api/consumer/signup 1.22 s

존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다 passed 1.22 s

Generated by IntelliJ IDEA on 5/9/24, 3:55 PM

모든 단위 테스트 결과

Tests in 'wiredcommerce': 2 total, 1 failed, 1 passed 1.13 s

Collapse | Expand

POST /api/consumer/signup		1.13 s
존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다	passed	1.11 s
올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다	failed	26 ms

Generated by IntelliJ IDEA on 5/9/24, 3:56 PM

테스트 실행 기록

```
2024-05-09T15:56:38.491+09:00 WARN 14968 --- [o-auto-1-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : SQL Error: 23505, SQLState: 23505
2024-05-09T15:56:38.491+09:00 ERROR 14968 --- [o-auto-1-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : Unique index or primary key violation: "PUBLIC.CONSTRAINT_INDEX_A ON PUBLIC.CONSUMER_ENTITY(EMAIL NULLS FIRST) VALUES ( /* 1 */ 'user@test.com' )"; SQL statement:
insert into consumer_entity (email,id) values (?,?) [23505-224]
```

거짓(False)양성(Positive)

거짓(False)양성(Positive)

테스트 실패

거짓(False) 양성(Positive)

올바른 운영 코드

테스트 실패

거짓양성 이유

```
@Test
void 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    var command = new SignUp("user@test.com", "my password");
    …
}

@Test
void 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    String email = "user@test.com";
    …
}
```

거짓양성 해결

```
@Test
void 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    var command = new SignUp(UUID.randomUUID() + "@test.com", "my password");
    …
}

@Test
void 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    String email = UUID.randomUUID() + "@test.com";
    …
}
```

거짓양성 해결 후 모든 단위 테스트 결과

PostTests: 2 total, 2 passed 1.16 s

Collapse | Expand

POST /api/signup		1.16 s
존재하는_이메일을_사용해_요청하면_400_상태코드를_반환한다	passed	1.14 s
올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다	passed	16 ms

Generated by IntelliJ IDEA on 5/6/24, 11:29 PM

비즈니스 규칙

판매자 엔터티

```
public class SellerEntity {

    private Long id;

    private String email;

    private String username;

    private String encodedPassword;

    private String phoneNumber;
}
```

이메일 주소

```
public class SellerEntity {

    private Long id;

    private String email;

    private String username;

    private String encodedPassword;

    private String phoneNumber;
}
```

전화번호

```
public class SellerEntity {

    private Long id;

    private String email;

    private String username;

    private String encodedPassword;

    private String phoneNumber;
}
```

GetProductsOfSeller

```
public record GetProductsOfSeller(Long sellerId) { }
```

GetProductsOfSellerQueryProcessor

```
package wiredcommerce.querymodel;

@AllArgsConstructor
public class GetProductsOfSellerQueryProcessor {

    private final EntityManager entityManager;

    public List<ProductView> process(GetProductsOfSeller query) {
        …
    }
}
```

상품 뷰

```
public record ProductView(
    UUID id,
    SellerView seller,
    String name,
    String description,
    double price,
    int stockQuantity
) { }
```

```
public record SellerView(
    Long id,
    String username
) { }
```

다른 판매자가 소유한 상품은 반환하지 않는다

```
@Test
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
)
```

임의의 데이터를 사용해 판매자 엔터티 생성

```
var random = new Random();

SellerEntity seller = SellerEntity
    .builder()
    .email(UUID.randomUUID() + "@test.com")
    .username(UUID.randomUUID().toString())
    .encodedPassword("password")
    .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
    .build();

sellerRepository.save(seller);
```

판매자가 소유한 상품 생성

```
productRepository.addProduct(new Product(
    UUID.randomUUID(),
    seller.getId(),
    "상품",
    "상품 설명",
    10000,
    100
));
```

임의의 데이터를 사용해 다른 판매자 엔터티 생성

```
SellerEntity anotherSeller = SellerEntity
    .builder()
    .email(UUID.randomUUID() + "@test.com")
    .username(UUID.randomUUID().toString())
    .encodedPassword("password")
    .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
    .build();

sellerRepository.save(anotherSeller);
```

다른 판매자가 소유한 상품 생성

```
productRepository.addProduct(new Product(
    UUID.randomUUID(),
    anotherSeller.getId(),
    "다른 상품",
    "다른 상품 설명",
    20000,
    200
));
```

조회 기능 실행과 결과 검증

```
var sut = new GetProductsOfSellerQueryProcessor(entityManager);

var query = new GetProductsOfSeller(seller.getId());
List<ProductView> actual = sut.process(query);

assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
```

다른 판매자가 소유한 상품은 반환하지 않는다

```
@Test
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    var random = new Random();

    SellerEntity seller = SellerEntity
        .builder()
        .email(UUID.randomUUID() + "@test.com")
        .username(UUID.randomUUID().toString())
        .encodedPassword("password")
        .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
        .build();

    sellerRepository.save(seller);

    productRepository.addProduct(new Product(
        UUID.randomUUID(),
        seller.getId(),
        "상품",
        "상품 설명",
        10000,
        100
    ));

    SellerEntity anotherSeller = SellerEntity
        .builder()
        .email(UUID.randomUUID() + "@test.com")
        .username(UUID.randomUUID().toString())
        .encodedPassword("password")
        .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000
        .build();

    sellerRepository.save(anotherSeller);

    productRepository.addProduct(new Product(
        UUID.randomUUID(),
        anotherSeller.getId(),
        "다른 상품",
        "다른 상품 설명",
        20000,
        200
    ));

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);

    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId());

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId(
}
```

AutoParams

https://github.com/AutoParams/AutoParams

패스트캠퍼스 The RED TDD 강의

The RED : 이규원의 현실 세상의 TDD : 안정감을 주는...

Top-tier TDD 프로그래밍

2021년 제작

The RED : 현실 세상의 TDD 실전편 : 설계 확장성을 위한...

TDD 테스트주도개발 더레드

2022년 제작

최근 커리어

2019년 2월

- 트립스토어 CTO
- Kotlin, Java, C#

2021년 3월

- 강남언니 CTO
- Java, Kotlin, C#

2023년 11월

- 와이어드컴퍼니 CTO
- TypeScript, Java

AutoParams 프로젝트 시작

Initial commit

Browse files

main

core-8.3.0 ... 0.0.1

gyuwon committed on Mar 6, 2021 Verified

0 parents commit 485efa

Publish 0.0.1

Browse files

main

core-8.3.0 ... 0.0.1

gyuwon committed on Mar 13, 2021

1 parent 5b60d71 commit cfc530

AutoParams 주 모듈 설치

```
dependencies {
    …
    testImplementation("io.github.autoparams:autoparams:8.3.0")
}
```

임의의 이메일 주소

```
@Test
void 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    var command = new SignUp(UUID.randomUUID() + "@test.com", "my password");
    …
}

@Test
void 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다() {
    String path = "/api/consumer/signup";
    String email = UUID.randomUUID() + "@test.com";
    …
}
```

@AutoSource를 사용한 임의 테스트 데이터 생성

```
@ParameterizedTest
@AutoSource
void 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다(SignUp signUp) {
    String path = "/api/consumer/signup";
    ResponseEntity<Void> response = client.postForEntity(path, signUp, Void.class);
    assertThat(response.getStatusCode().is2xxSuccessful()).isTrue();
}
```

@AutoSource를 사용한 임의 테스트 데이터 생성

```
@ParameterizedTest
@AutoSource
void 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다(SignUp signUp, String otherPassword) {
    String path = "/api/consumer/signup";
    client.postForEntity(path, signUp, Void.class);

    ResponseEntity<Void> response = client.postForEntity(
        path,
        new SignUp(signUp.email(), otherPassword),
        Void.class
    );

    assertThat(response.getStatusCode().value()).isEqualTo(400);
}
```

@AutoSource를 사용한모든 단위 테스트 결과

Tests in 'wiredcommerce': 2 total, 2 passed 1.16 s

Collapse | Expand

POST /api/consumer/signup 1.16 s

- 존재하는_이메일_주소를_사용해_요청하면_400_상태코드를_반환한다 1.15 s
  - [1] signUp=SignUp[email=e3321df4-64d3-472c-87e6-6e659b651ed5@test.com, password=79f60de6-ad94-4219-8988-acfcdb43da08], otherPassword=81399fd6-dfd7-433f-9600-87679f27d9b3 passed 1.15 s
- 올바른_정보를_사용해_요청하면_성공_상태코드를_반환한다 13 ms
  - [1] signUp=SignUp[email=015105d6-84de-4b4e-984e-3b744da49d36@test.com, password=bcc52c3f-76d8-46fe-9f2b-7f195dc28c2f] passed 13 ms

Generated by IntelliJ IDEA on 5/9/24, 4:21 PM

다른 판매자가 소유한 상품은 반환하지 않는다

```
@Test
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    var random = new Random();

    SellerEntity seller = SellerEntity
        .builder()
        .email(UUID.randomUUID() + "@test.com")
        .username(UUID.randomUUID().toString())
        .encodedPassword("password")
        .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
        .build();

    sellerRepository.save(seller);

    productRepository.addProduct(new Product(
        UUID.randomUUID(),
        seller.getId(),
        "상품",
        "상품 설명",
        10000,
        100
    ));

    SellerEntity anotherSeller = SellerEntity
        .builder()
        .email(UUID.randomUUID() + "@test.com")
        .username(UUID.randomUUID().toString())
        .encodedPassword("password")
        .phoneNumber("010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000
        .build();

    sellerRepository.save(anotherSeller);

    productRepository.addProduct(new Product(
        UUID.randomUUID(),
        anotherSeller.getId(),
        "다른 상품",
        "다른 상품 설명",
        20000,
        200
    ));

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);

    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId());

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId(
}
```

다른 판매자가 소유한 상품은 반환하지 않는다

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

@AutoSource 확장

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

임의의 판매자 엔터티 제공

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

다른 임의의 판매자 엔터티 제공

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

상품 엔터티 생성기

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

스프링 의존성 주입

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

판매자 엔터티 저장

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

상품 엔터티

```
public record Product(
    UUID id,
    long sellerId,
    String name,
    String description,
    int price,
    int stockQuantity
) {
}
```

상품 엔터티 생성기에 판매자 식별자 고정

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

이후 factory가 Product 엔터티 생성시 sellerId 속성을 **seller** 엔터티의 id 필드로 설정

AutoParams API를 사용한 기능 확장

상품 생성

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

다른 판매자 엔터티 저장

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

상품 엔터티 생성기에 다른 판매자 식별자 고정

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

이후 factory가 Product 엔터티 생성시 sellerId 속성을 **anotherSeller** 엔터티의 id 필드로 설정

상품 생성

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

GetProductsOfSeller 조회 실행

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

조회 결과 상품이 모두 seller의 것인지 검사

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

```
public record ProductView(
    UUID id,
    SellerView seller,
    String name,
    String description,
    double price,
    int stockQuantity
) { }
```

```
public record SellerView(
    Long id,
    String username
) { }
```

내부 핵심 아키텍처

class ResolutionContext

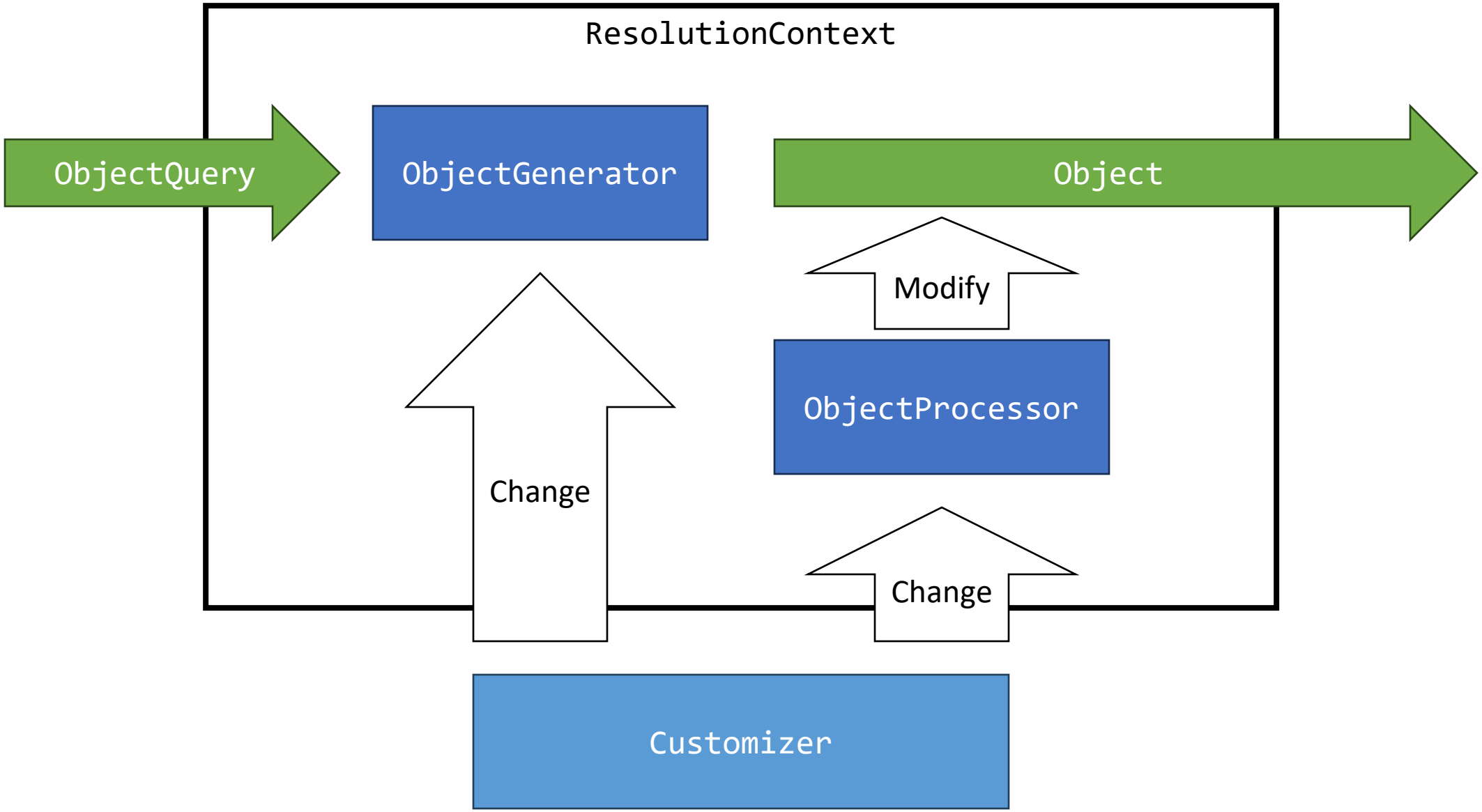
```
Object resolve(ObjectQuery query)
void applyCustomizer(Customizer customizer)
```

ResolutionContext.resolve

```
public Object resolve(ObjectQuery query) {
    Object value = generateValue(query);
    processValue(query, value);
    return value;
}
```

ResolutionContext.applyCustomizer

```
public void applyCustomizer(Customizer customizer) {
    generator = customizer.customize(generator);
    processor = customizer.customize(processor);
}
```


ResolutionContext
ObjectQuery
ObjectGenerator
Object
Modify
ObjectProcessor
Change
Change
Customizer

interface ObjectQuery

```
Type getType();
```

```
class TypeQuery implements ObjectQuery
```

class ParameterQuery implements ObjectQuery

```
Parameter getParameter()
```

class ObjectContainer

- Object unwrapOrElseThrow()
  ObjectContainer process(Function<Object, Object> processor)
  ObjectContainer yieldIfEmpty(Supplier<ObjectContainer> next)

class ObjectContainer

```
Object unwrapOrElseThrow()
ObjectContainer process(Function<Object, Object> processor)
ObjectContainer yieldIfEmpty(Supplier<ObjectContainer> next)
```

class ObjectContainer

```
Object unwrapOrElseThrow()
ObjectContainer process(Function<Object, Object> processor)
ObjectContainer yieldIfEmpty(Supplier<ObjectContainer> next)
```

interface ObjectGenerator

```
ObjectContainer generate(ObjectQuery query, ResolutionContext context)
```

freezeSellerId

```
public record SellerIdFreezer(long sellerId) implements ObjectGenerator {

    public static SellerIdFreezer freezeSellerId(SellerEntity seller) {
        return new SellerIdFreezer(seller.getId());
    }

    public ObjectContainer generate(ObjectQuery query, ResolutionContext context) {
        return query.getType().equals(long.class)
            && query instanceof ParameterQuery parameterQuery
            ? generate(parameterQuery)
            : ObjectContainer.EMPTY;
    }

    private ObjectContainer generate(ParameterQuery query) {
        return query
            .getParameterName()
            .filter(name -> name.equals("sellerId"))
            .map(name -> new ObjectContainer(sellerId))
            .orElse(ObjectContainer.EMPTY);
    }
}
```

freezeSellerId 함수 사용

```
@ParameterizedTest
@AutoDomainSource
void 다른_판매자가_소유한_상품은_반환하지_않는다(
    SellerEntity seller,
    SellerEntity anotherSeller,
    Factory<Product> factory,
    @Autowired SellerJpaRepository sellerRepository,
    @Autowired ProductJpaRepository productRepository,
    @Autowired EntityManager entityManager
) {
    sellerRepository.save(seller);
    factory.applyCustomizer(freezeSellerId(seller));
    productRepository.addProduct(factory.get());

    sellerRepository.save(anotherSeller);
    factory.applyCustomizer(freezeSellerId(anotherSeller));
    productRepository.addProduct(factory.get());

    var sut = new GetProductsOfSellerQueryProcessor(entityManager);
    List<ProductView> actual = sut.process(new GetProductsOfSeller(seller.getId()));

    assertThat(actual).extracting(x -> x.seller().id()).containsExactly(seller.getId());
}
```

interface ObjectProcessor

```
void process(ObjectQuery query, Object value, ResolutionContext context)
```

interface Customizer

```
ObjectGenerator customize(ObjectGenerator generator)
ObjectProcessor customize(ObjectProcessor processor)
```

interface Customizer

```
ObjectGenerator customize(ObjectGenerator generator)
ObjectProcessor customize(ObjectProcessor processor)

interface ObjectGenerator extends Customizer
interface ObjectProcessor extends Customizer
```

@interface Customization

```
Class<? extends Customizer>[] value();
```

확장

@AutoDomainSource

```
@AutoSource
@AutoDomainSourceConfiguration
public @interface AutoDomainSource {
}
```

@AutoSource 기능 사용

```
@AutoSource
@AutoDomainSourceConfiguration
public @interface AutoDomainSource {
}
```

기능 확장 구성

```
@AutoSource
@AutoDomainSourceConfiguration
public @interface AutoDomainSource {
}
```

@AutoDomainSourceConfiguration

```
@Customization({
    PhoneNumberGenerator.class,
    PriceGenerator.class,
    QuantityGenerator.class,
})
@BrakeBeforeAnnotation(Autowired.class)
public @interface AutoDomainSourceConfiguration {
}
```

비즈니스 규칙을 반영한 개체 생성기 적용

```
@Customization({
    PhoneNumberGenerator.class,
    PriceGenerator.class,
    QuantityGenerator.class,
})
@BrakeBeforeAnnotation(Autowired.class)
public @interface AutoDomainSourceConfiguration {
}
```

PhoneNumberGenerator

```
public class PhoneNumberGenerator implements ObjectGenerator {

    @Override
    public ObjectContainer generate(ObjectQuery query, ResolutionContext context) {
        return query.getType().equals(String.class)
            && query instanceof ParameterQuery parameterQuery
            ? generator(parameterQuery)
            : ObjectContainer.EMPTY;
    }

    private ObjectContainer generator(ParameterQuery query) {
        return query
            .getParameterName()
            .map(String::toLowerCase)
            .filter(name -> name.endsWith("phonenumber"))
            .map(name -> ThreadLocalRandom.current())
            .map(random -> "010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
            .map(ObjectContainer::new)
            .orElse(ObjectContainer.EMPTY);
    }
}
```

PhoneNumberGenerator

```
public class PhoneNumberGenerator implements ObjectGenerator {

    @Override
    public ObjectContainer generate(ObjectQuery query, ResolutionContext context) {
        return query.getType().equals(String.class)
            && query instanc                    meterQuery
            ? generator(para
            : ObjectContaine
    }

    private ObjectContainer generator(ParameterQuery query) {
        return query
            .getParameterName()
            .map(String::toLowerCase)
            .filter(name -> name.endsWith("phonenumber"))
            .map(name -> ThreadLocalRandom.current())
            .map(random -> "010-" + random.nextInt(1000, 10000) + "-" + random.nextInt(1000, 10000))
            .map(ObjectContainer::new)
            .orElse(ObjectContainer.EMPTY);
    }
}
```

Optional<String>

-parameters

https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javac.html

Stores formal parameter names of constructors and methods in the generated class file so that the method java.lang.reflect.Executable.getParameters from the Reflection API can retrieve them.

https://docs.spring.io/spring-boot/maven-plugin/using.html#using

Maven users can inherit from the spring-boot-starter-parent project to obtain sensible defaults. The parent project provides the following features:

…

- Compilation with -parameters.

…

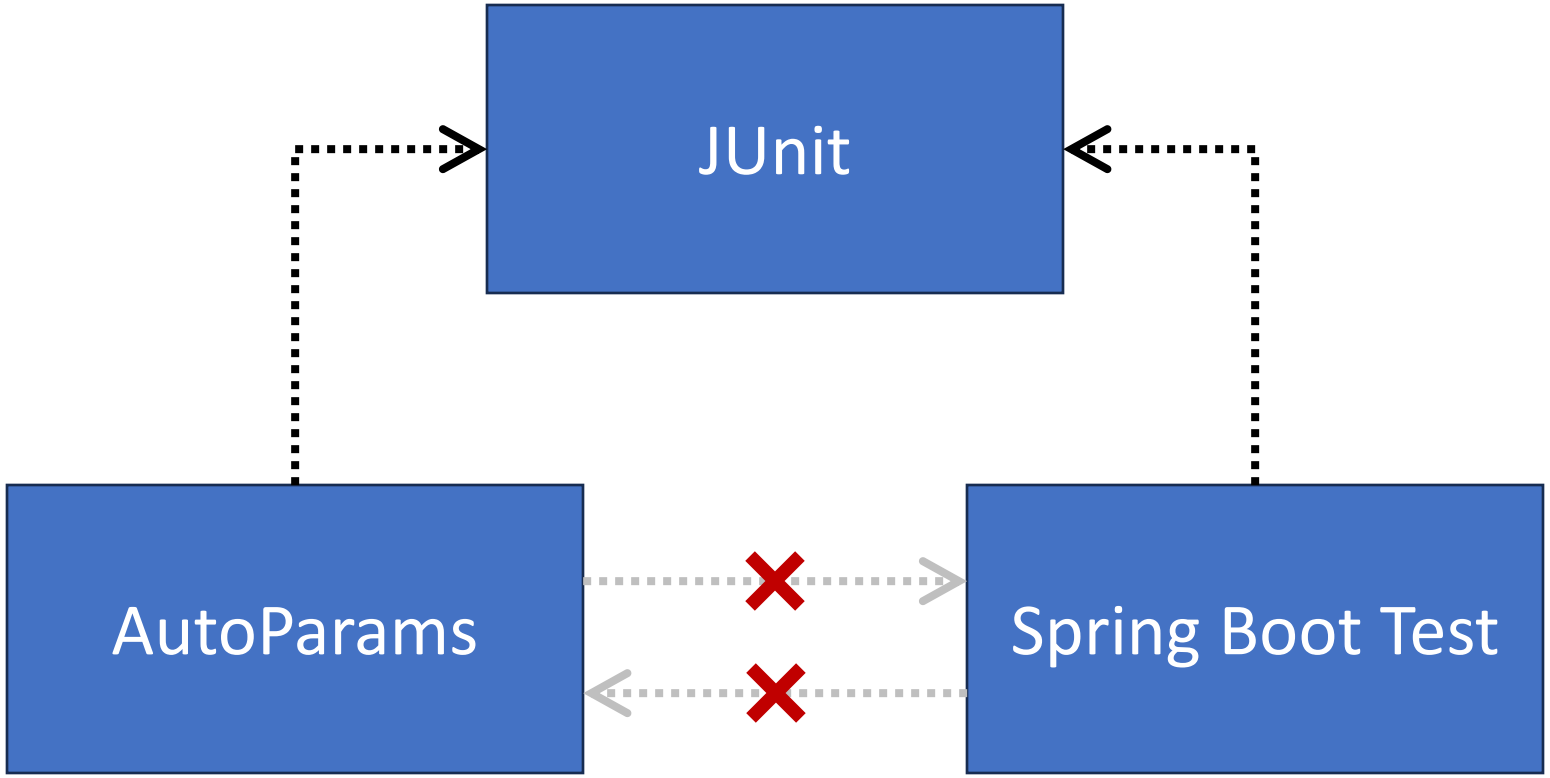
AutoParams * Spring Boot Test

```
@Customization({
    PhoneNumberGenerator.class,
    PriceGenerator.class,
    QuantityGenerator.class,
})
@BrakeBeforeAnnotation(Autowired.class)
public @interface AutoDomainSourceConfiguration {
}
```

@BrakeBeforeAnnotation(Autowired.class)가 없다면

매개변수 해결 경쟁

```
Discovered multiple competing ParameterResolvers for parameter
[org.springframework.boot.test.web.client.TestRestTemplate client] in
method [void
test.wiredcommerce.api.seller.signup.PostTests.존재하는_이메일_주소를_사용해
_요청하면_400_상태코드를_반환한다(wiredcommerce.seller.command.SignUp,java.l
ang.String,java.lang.String,java.lang.String,org.springframework.boot.tes
t.web.client.TestRestTemplate)]:
org.springframework.test.context.junit.jupiter.SpringExtension@1da77a43,
org.junit.jupiter.params.ParameterizedTestParameterResolver@63604641
```


JUnit
AutoParams
Spring Boot Test

AutoParams * Spring Boot Test

```
@Customization({
    PhoneNumberGenerator.class,
    PriceGenerator.class,
    QuantityGenerator.class,
})
@autoparams.BrakeBeforeAnnotation(org.springframework.beans.factory.annot
ation.Autowired.class)
public @interface AutoDomainSourceConfiguration {
}
```

그 밖의 주요 기능

- **`@ValueAutoSource`** → `@ValueSource` * `@AutoSource`
- **`@EnumAutoSource`** → `@EnumSource` * `@AutoSource`
- **`@CsvAutoSource`** → `@CsvSource` * `@AutoSource`
- **`@MethodAutoSource`** → `@MethodSource` * `@AutoSource`
- **`@Repeat`** → 반복 실행
- **`@Freeze`** → 매개변수 고정
- **`InstancePropertyWriter`** → Setter 지원
- **`InstanceFieldWriter`** → 필드 지원
- **autoparams-kotlin** → Kotlin 지원
- **autoparams-lombok** → Lombok 지원
- **autoparams-mockito** → Mockito 지원

질문과 답변