Kotlin + Spring Data JPA

김태호

안녕하세요

김태호

VCNC

sapzil.org (블로그)
github.com/dittos


ミトニト

타다 프로젝트

2018년 6월~ (10개월)

Kotlin + Spring Boot + JPA + ...

Kotlin 코드 약 6만 줄

JPA 엔티티 36종

이런 분들을 위한 발표입니다

Spring Data JPA: 사용해 봄 (Java에서)

Kotlin: 전혀 모름 ~ 약간 사용해 봄

목차

그래서 Kotlin 쓸만한가요?

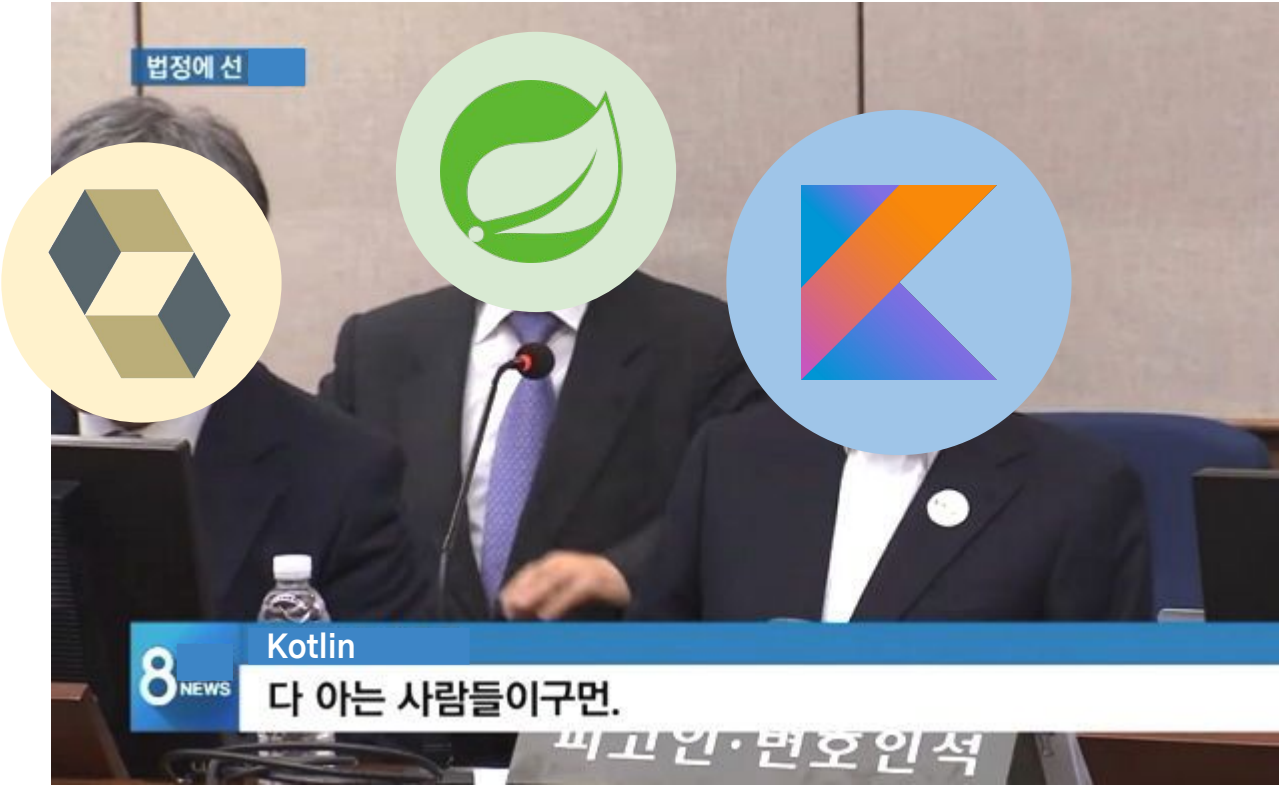
null 안전성

간결한 코드

```
final List<String> greetings =
  people.stream()
    .map(it -> "Hello " + it.getName() + "!")
    .collect(Collectors.toList());
```

```
val greetings =
  people.map { "Hello ${it.name}!" }
```

Java 호환



Kotlin ↔ Java 호환성

Kotlin에서 Java 코드 호출

Java에서 Kotlin 코드 호출

Kotlin에서 Java 코드 호출 😅

비교적 자연스럽게 사용 가능

Kotlin은 Java를 고려하여 설계됨

Java에서 Kotlin 코드 호출 🤔

언제나 자연스럽게 되지는 않음

기존 Java 코드는 Kotlin에 대해 모름

Java 쪽에서 Kotlin 클래스가 어떻게 '보이는' 지가 중요

특히 리플렉션을 활용하는 경우 (JPA!)

목차

Repository (1)

```java
public interface UserRepository extends CrudRepository<User, Long> {
   User findByUsername(String username);
}
```

```fun
interface UserRepository : CrudRepository<User, Long> {
   fun findByUsername(username: String): User?
}
```

Repository (1)

```java
public interface UserRepository extends CrudRepository<User, Long> {
   User findByUsername(String username);
}
```

```fun
interface UserRepository : CrudRepository<User, Long> {
   fun findByUsername(username: String): User?
}
```

Java

T

nullable

≠

Kotlin

T

not nullable

Java

T

nullable

Kotlin

T?

nullable

Repository (1)

```
interface UserRepository : CrudRepository<User, Long> {
    fun findByUsername(username: String): User
}
```

```
personRepository.findByUsername("nobody")
```

Repository (1)

```
interface UserRepository : CrudRepository<User, Long> {
   fun findByUsername(username: String): User
}
```

```
personRepository.findByUsername("nobody")
org.springframework.dao.EmptyResultDataAccessException: Result must
not be null!
    at org.springframework.data.repository.core.support.MethodInvocationValidator.invoke
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
    ...
```

Repository (1)

```
interface UserRepository : CrudRepository<User, Long> {
    fun findByUsername(username: String): User
}
```

nullability에 주의!

```
personRepository.findByUsername("nobody")
```

```
org.springframework.dao.EmptyResultDataAccessException: Result must
not be null!
    at org.springframework.data.repository.core.support.MethodInvocationValidator.invoke
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
```

https://docs.spring.io/spring-data/jpa/docs/2.1.6.RELEASE/reference/html/#repositories.nullability.kotlin

Repository (2)

```
val optionalUser: Optional<User> = userRepository.findById(1)


optionalUser.map { it.username }.orElse("")
optionalUser.orElse(null)?.username ?: ""
```

Java Optional은 Kotlin에서 불편합니다.

Spring Data Commons / DATACMNS-1346

Add a findByIdOrNull(…) Kotlin extension to CrudRepository

Comment | Agile Board | Watch issue

Details

Type:	Improvement	Status:	CLOSED
Priority:	Minor	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	2.2 M1 (Moore), ... (1)
Component/s:	Repositories		
Labels:	kotlin		
Pull Request URL:	https://github.com/spring-projects/spring-data-commons/pull/299		

Description

In Kotlin, it is idiomatic to deal with return value that could have or not a result with nullable types since they are natively supported by the language. This commit add a `findByIdOrNull` variant to `CrudRepository#findById` that returns `T?` instead of `Optional<T>`.

https://jira.spring.io/browse/DATACMNS-1346

Spring Data Commons / DATACMNS-1346

Add a findByIdOrNull(…) Kotlin extension to CrudRepository

Comment | Agile Board | Watch issue

Details

Type:	Improvement	Status:	CLOSED
Priority:	Minor	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	2.2 M1 (Moore), ... (1)
Component/s:	Repositories		
Labels:	kotlin		
Pull Request URL:	https://github.com/spring-projects/spring-data-commons/pull/299		

Description

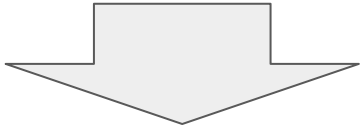
In Kotlin, it is idiomatic to deal with return value that could have or not a result with nullable types since they are natively supported by the language. This commit add a `findByIdOrNull` variant to `CrudRepository#findById` that returns `T?` instead of `Optional<T>`.

Spring Data **2.1.4**에 추가
(= Spring Boot 2.1.2)

https://jira.spring.io/browse/DATACMNS-1346

findByIdOrNull

```
val optionalUser: Optional<User> = userRepository.findById(1)


optionalUser.map { it.username }.orElse("")
optionalUser.orElse(null)?.username ?: ""
```



```
val user: User? = userRepository.findByIdOrNull(1)
user?.username ?: ""
```

findByIdOrNull

```
import org.springframework.data.repository.findByIdOrNull


val user: User? = userRepository.findByIdOrNull(1)
user?.username ?: ""
```

Kotlin extension function으로 구현되어 **import 필요**

목차

Entity: Java에서

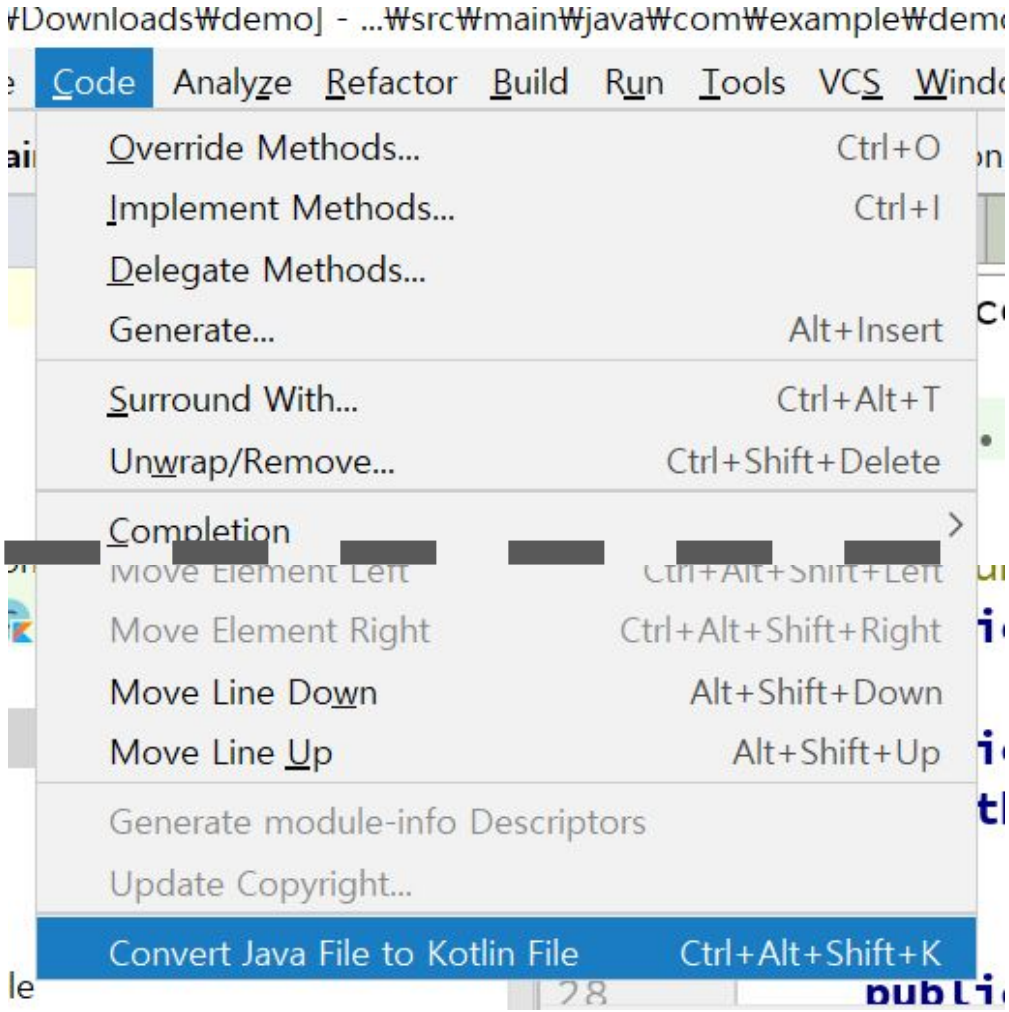
```java
@Entity
public class Person {
   @Id @GeneratedValue
   private Long id;

   @Column(nullable = false)
   private String name;

   // optional
   private String phoneNumber;

   // getters, setters
}
```

```java
@Entity
public class Person {
   private Long id;
   private String name;
   private String phoneNumber;

   @Id @GeneratedValue
   public Long getId() { return id; }

   @Column(nullable = false)
   public String getName() { return name; }

   public String getPhoneNumber() { return
phoneNumber; }

   // setters...
}
```

Kotlin으로 바꿔봅시다

IntelliJ 기능을 활용

Code – Convert Java File to Kotlin File



Kotlin으로 바꿔봅시다

```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String? = null

    var phoneNumber: String? = null
}
```

```kotlin
@Entity
class Person {
    @get:Id
    @get:GeneratedValue
    var id: Long? = null

    @get:Column(nullable = false)
    var name: String? = null

    var phoneNumber: String? = null
}
```

Kotlin으로 바꿔봅시다

```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String? = null

    var phoneNumber: String? = null
}
```

Kotlin property는
명시적 초기화 필요

Java에서는 어떻게 보일까요?

Tools – Kotlin – Show Kotlin Bytecode

Kotlin Bytecode

Decompile ☑ Inline ☑ Optimization ☑ Assertions ☐ IR ☐ JVM 8 target

```
// ================com/example/demo/Person.class =================
// class version 50.0 (50)
// access flags 0x31
public final class com/example/demo/Person {


  @Ljavax/persistence/Entity;()

  // access flags 0x2
  private Ljava/lang/Long; id
  @Ljavax/persistence/Id;()
  @Ljavax/persistence/GeneratedValue;()
  @Lorg/jetbrains/annotations/Nullable;() // invisible

  // access flags 0x11
  public final getId()Ljava/lang/Long;
```

Ant Build | Maven | Gradle | Kotlin Bytecode

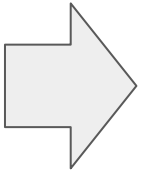
```
@Entity
@Metadata(
    mv = {1, 1, 15},
    bv = {1, 0, 3},
    k = 1,
    d1 = {"\u0000\u001c\n\u0002\u0018\u0002\n\u0002\u0010\u0000\r
    d2 = {"Lcom/example/demo/Person;", "", "()V", "id", "", "getI
)
public final class Person {
    @Id
    @GeneratedValue
    @Nullable
    private Long id;
    @Column(
        nullable = false
    )
    @Nullable
    private String name;
    @Nullable
    private String phoneNumber;
```

```
@Entity
@Metadata(
    mv = {1, 1, 15},
    bv = {1, 0, 3},
    k = 1,
    d1 = {"\u0000\u001c\n\u0002\u0018\u0002\n\u0002\u0010\u0000\r
    d2 = {"Lcom/example/demo/Person;", "", "()V", "id", "", "getI
)
public final class Person {
    @Id
    @GeneratedValue
    @Nullable
    private Long id;
    @Column(
        nullable = false
    )
    @Nullable
    private String name;
    @Nullable
    private String phoneNumber;
```

```java
@Nullable
public final Long getId() { return this.id; }

public final void setId(@Nullable Long var1) { this.id = var1; }

@Nullable
public final String getName() { return this.name; }

public final void setName(@Nullable String var1) { this.name = var1; }

@Nullable
public final String getPhoneNumber() { return this.phoneNumber; }

public final void setPhoneNumber(@Nullable String var1) { this.phoneNumber = var1; }
```

좀 더 Kotlin스럽게: Non-nullable Type

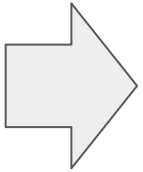
```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String? = null

    var phoneNumber: String? = null
}
```



```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String = ""

    var phoneNumber: String? = null
}
```

좀 더 Kotlin스럽게: Named Arguments

```kotlin
val person = Person()
person.id = 1
person.name = "hi"
person.phoneNumber = "1234"
```

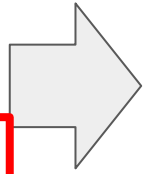


```kotlin
val person = Person(
  id = 1,
  name = "hi",
  phoneNumber = "1234"
)
```

```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String = ""

    var phoneNumber: String? = null

    constructor()

    constructor(id: Long?, name: String, phoneNumber: String?) {
        this.id = id
        this.name = name
        this.phoneNumber = phoneNumber
    }
}
```

좀 더 Kotlin스럽게: Primary Constructor

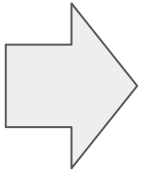
```kotlin
@Entity
class Person {
    @Id
    @GeneratedValue
    var id: Long? = null

    @Column(nullable = false)
    var name: String = ""

    var phoneNumber: String? = null

    constructor()

    constructor(id: Long?, name: String,
phoneNumber: String?) {
        this.id = id
        this.name = name
        this.phoneNumber = phoneNumber
    }
}
```



```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long? = null,

    @Column(nullable = false)
    var name: String = "",

    var phoneNumber: String? = null
)
```

좀 더 Kotlin스럽게: 기본값 없애기

```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long? = null,

    @Column(nullable = false)
    var name: String = "",

    var phoneNumber: String? = null
)
```

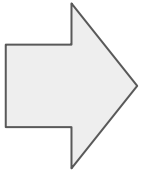


```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```

좀 더 Kotlin스럽게: 기본값 없애기

```
org.springframework.orm.jpa.JpaSystemException:
No default constructor for entity:  :
com.example.demo.Person
```

```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long? = null,

    @Column(nullable = false)
    var name: String = "",

    var phoneNumber: String? = null
)
```



```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```

kotlin-noarg (kotlin-jpa)

Kotlin 컴파일러 플러그인 ✨

특정 어노테이션이 붙은 클래스에 no-arg constructor를 자동으로 만들어줍니다.

Gradle/Maven 플러그인으로 추가합니다.

kotlin-jpa

kotlin-noarg + JPA를 위한 기본 설정

`@Entity`, `@Embeddable`, `@MappedSuperclass`

http://kotlinlang.org/docs/reference/compiler-plugins.html#jpa-support

```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```

feat. kotlin-jpa 플러그인

번외편: data class?

다음 메소드를 자동으로 구현:

`equals() / hashCode()`

`toString()`

`copy()`

Data class

```
@Entity
data class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```

Data class

```
@Entity
data class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```

equals/hashCode를 호출하게 될 때 주의 필요

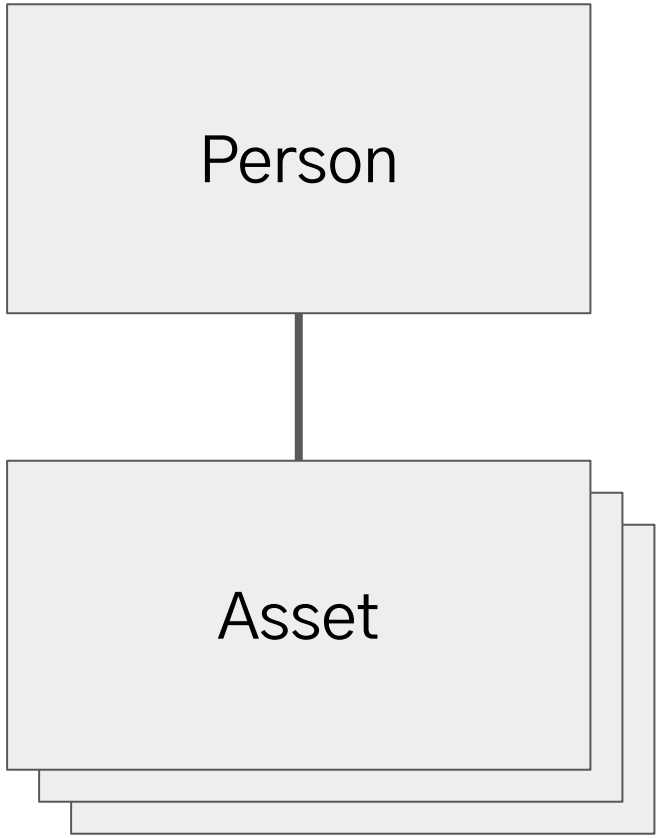
==, toSet()

순환 참조가 있으면 무한 재귀 호출에 빠짐

목차

@ManyToOne과 지연 로딩

```
@Entity
class Asset(
    @Id
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    @ManyToOne(fetch = FetchType.LAZY)
    var person: Person
)
```



지연 로딩: 기대

```
val a = assetRepository.findByIdOrNull(1)!!
```

```
SQL: select ... from asset asset0_ where asset0_.id=?
```

Person에 대한 쿼리 X

지연 로딩: 기대

```
val a = assetRepository.findByIdOrNull(1)!!

val person = a.person
println(person::class)
```

```
class com.example.demo.Person$HibernateProxy$lsHeDMGk
```

프록시 객체

Person에 대한 쿼리 X

지연 로딩: 기대

```
val a = assetRepository.findByIdOrNull(1)!!

val person = a.person
println(person.id)
```

Person에 대한 쿼리 X (Asset만으로 알 수 있음)

지연 로딩: 기대

```
val a = assetRepository.findByIdOrNull(1)!!

val person = a.person
println(person.id)
println(person.name)
```

```
SQL: select ... from person person0_ where person0_.id=?
```

지연 로딩

지연 로딩: 실제

```
val a = assetRepository.findByIdOrNull(1)!!
```

```
SQL: select ... from asset asset0_ where asset0_.id=?
SQL: select ... from person person0_ where person0_.id=?
```

지연 로딩: 실제

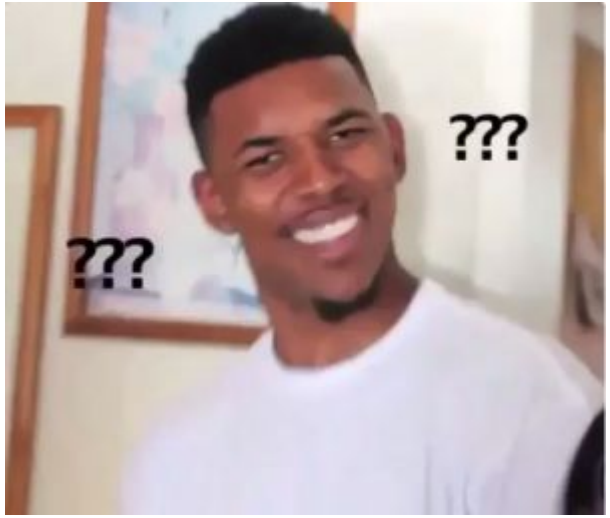
```
val a = assetRepository.findByIdOrNull(1)!!
```

```
SQL: select ... from asset asset0_ where asset0_.id=?
SQL: select ... from person person0_ where person0_.id=?
```

```
val person = a.person
println(person::class)
```

```
class com.example.demo.Person
```

프록시 객체가 아니다?



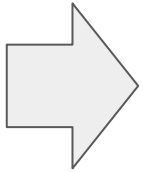
Final by default

Kotlin 클래스는 final (상속 불가)이 기본

프록시 클래스를 생성하려면 클래스가 상속 가능해야 합니다.

open! open! open!

```
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```



```
@Entity
open class Person(
    @Id
    @GeneratedValue
    open var id: Long?,

    @Column(nullable = false)
    open var name: String,

    open var phoneNumber: String?
)
```

kotlin-allopen

Kotlin 컴파일러 플러그인 ✨

특정 어노테이션이 붙은 클래스와 그 클래스의 멤버를 자동으로 open으로 만들어줍니다.

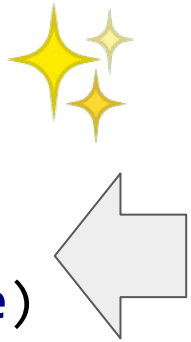
Gradle/Maven 플러그인으로 추가합니다.

```
allOpen {
    annotation("javax.persistence.Entity")
}
```

http://kotlinlang.org/docs/reference/compiler-plugins.html#all-open-compiler-plugin

kotlin-allopen

```kotlin
@Entity
class Person(
    @Id
    @GeneratedValue
    var id: Long?,

    @Column(nullable = false)
    var name: String,

    var phoneNumber: String?
)
```



```kotlin
@Entity
open class Person(
    @Id
    @GeneratedValue
    open var id: Long?,

    @Column(nullable = false)
    open var name: String,

    open var phoneNumber:
String?
)
```

목차

Kotlin 컬렉션

Immutable	Mutable
`List<T>`	`MutableList<T>`
`Set<T>`	`MutableSet<T>`
`Map<K, V>`	`MutableMap<K, V>`

Kotlin 컬렉션

Immutable	Mutable	Java
List<T>	MutableList<T>	java.util.List<T>
Set<T>	MutableSet<T>	java.util.Set<T>
Map<K, V>	MutableMap<K, V>	java.util.Map<K, V>

@OneToMany

```
@Entity
class Person(
    ...,

    @OneToMany
    var assets: List<Asset>
)
```

@OneToMany

```
@Entity
class Person(
    ...,

    @OneToMany
    var assets: List<Asset>
)
```

```
org.hibernate.AnnotationException:
Collection has neither generic type or
OneToMany.targetEntity() defined:
com.example.demo.Person.assets
```

```
// access flags 0x2
// signature Ljava/util/List<+Lcom/example/demo/Asset;>;
// declaration: assets extends java.util.List<? extends com.example.demo.Asset>
private Ljava/util/List; assets
@Ljavax/persistence/OneToMany;()
@Lorg/jetbrains/annotations/NotNull;() // invisible
```

@OneToMany

```
@Entity
class Person(
    ...,

    @OneToMany
    var assets: List<@JvmSuppressWildcards Asset>
)
```

@OneToMany

```
@Entity
class Person(
    ...,

    @OneToMany
    var assets: MutableList<Asset>
)
```

```
/**
 * A generic ordered collection of elements. Methods in this in
 * read/write access is supported through the [MutableList] int
 * @param E the type of elements contained in the list. The lis
 */
public interface List<out E> : Collection<E> {
    // Query Operations
    override val size: Int

    override fun isEmpty(): Boolean
    override fun contains(element: @UnsafeVariance E): Boolean
    override fun iterator(): Iterator<E>
```

Type Variance

Immutable	Mutable
`List<out T>`	`MutableList<T>`
`Set<out T>`	`MutableSet<T>`
`Map<K, out V>`	`MutableMap<K, V>`

Type Variance

Kotlin	Java
`List<out T>`	`java.util.List <? extends T>`
`Set<out T>`	`java.util.Set <? extends T>`
`Map<K, out V>`	`java.util.Map <K, ? extends V>`

https://kotlinlang.org/docs/reference/java-to-kotlin-interop.html#variant-generics

정리

똑같이 생겼지만 의미가 다른 코드를 조심하자

	Java	Kotlin
T	nullable	non-nullable
class	non-final	final
List<T>	mutable	immutable

정리

간결한 코드를 위해서 컴파일러 플러그인의 도움을 받을 수 있다

- kotlin-jpa, kotlin-allopen

Java 호환성 문제가 있을 때는 바이트코드를 확인해보자

감사합니다 🙇‍♂️

끝