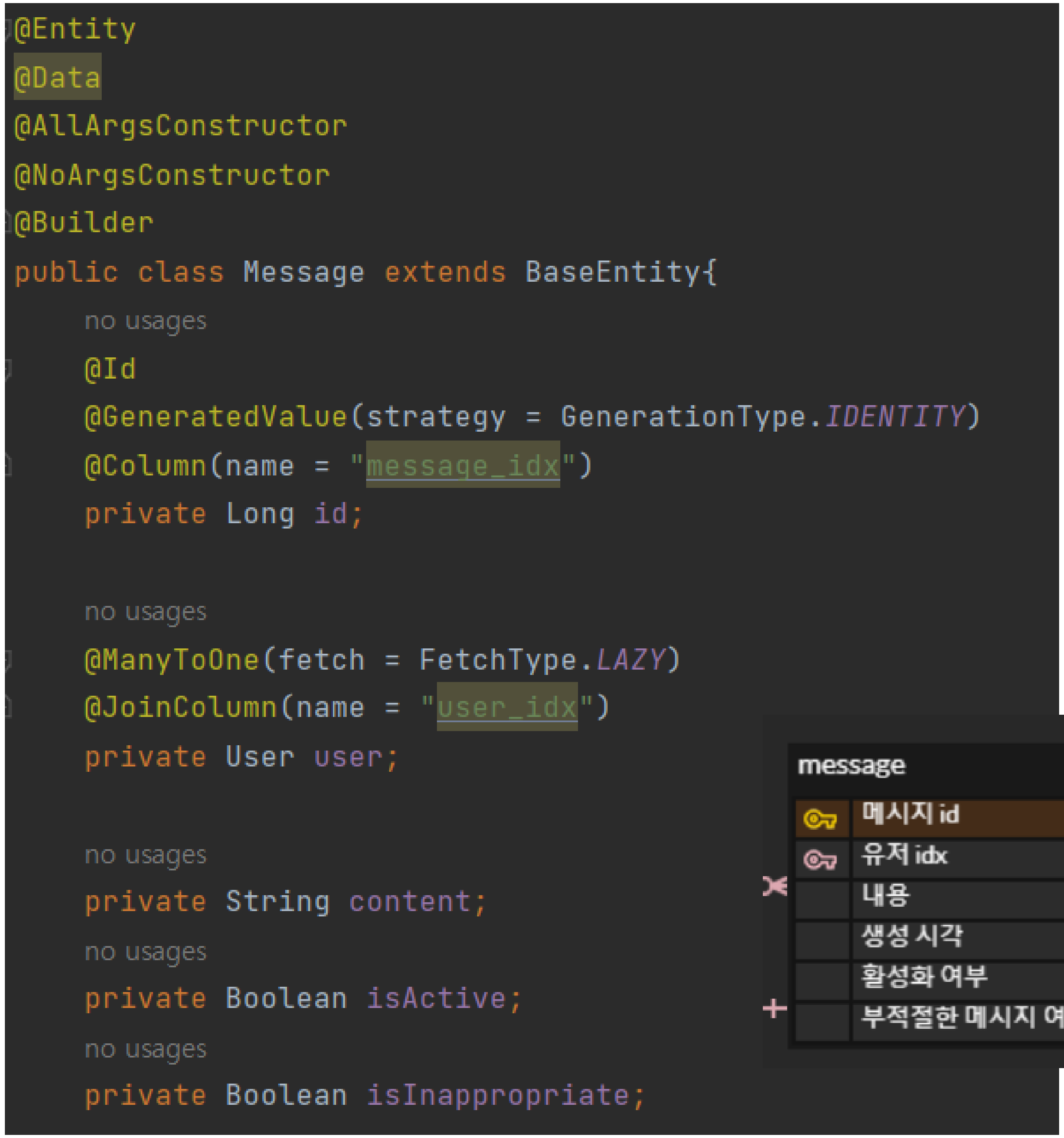
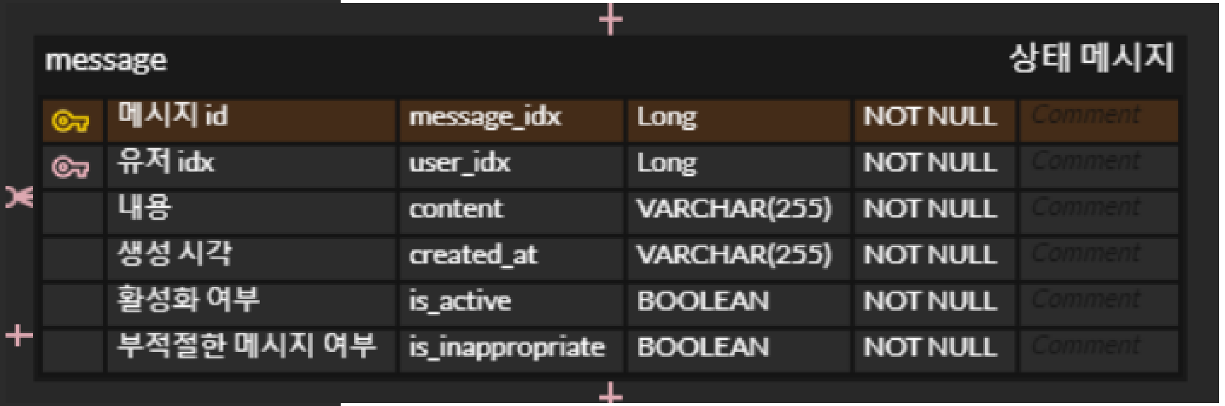
스프링 부트

$$\begin{tikzcd}
\text{Client} \arrow[r, leftrightarrow, "\text{DTO}"] & \text{Controller} \arrow[r, leftrightarrow, "\text{DTO}"] & \text{Service} \arrow[r, leftrightarrow, "\text{Entity}"] & \text{Repository} \arrow[r, leftrightarrow, "\text{Entity}"] & \text{DB}
\end{tikzcd}$$

Entity



```
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Message extends BaseEntity{
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "message_idx")
    private Long id;

    no usages
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_idx")
    private User user;

    no usages
    private String content;
    no usages
    private Boolean isActive;
    no usages
    private Boolean isInappropriate;
```



message — 상태 메시지

🔑	메시지 id	message_idx	Long	NOT NULL	Comment
🔑	유저 idx	user_idx	Long	NOT NULL	Comment
	내용	content	VARCHAR(255)	NOT NULL	Comment
	생성 시각	created_at	VARCHAR(255)	NOT NULL	Comment
	활성화 여부	is_active	BOOLEAN	NOT NULL	Comment
	부적절한 메시지 여부	is_inappropriate	BOOLEAN	NOT NULL	Comment

IoC (Inversion of Control)
제어의 역전

DI (Dependency Injection)
의존성 주입

```
class Chef {

        레시피


        요리하기() {
                레시피 단계 보기()
                단계 따라 요리하기()
        }

}
```

```
class Chef {

        BurgerRecipe burgerRecipe
            = new BurgerRecipe();

        cook() {
                burgerRecipe.getStep();
                // 버거 요리 하기
        }

}
```

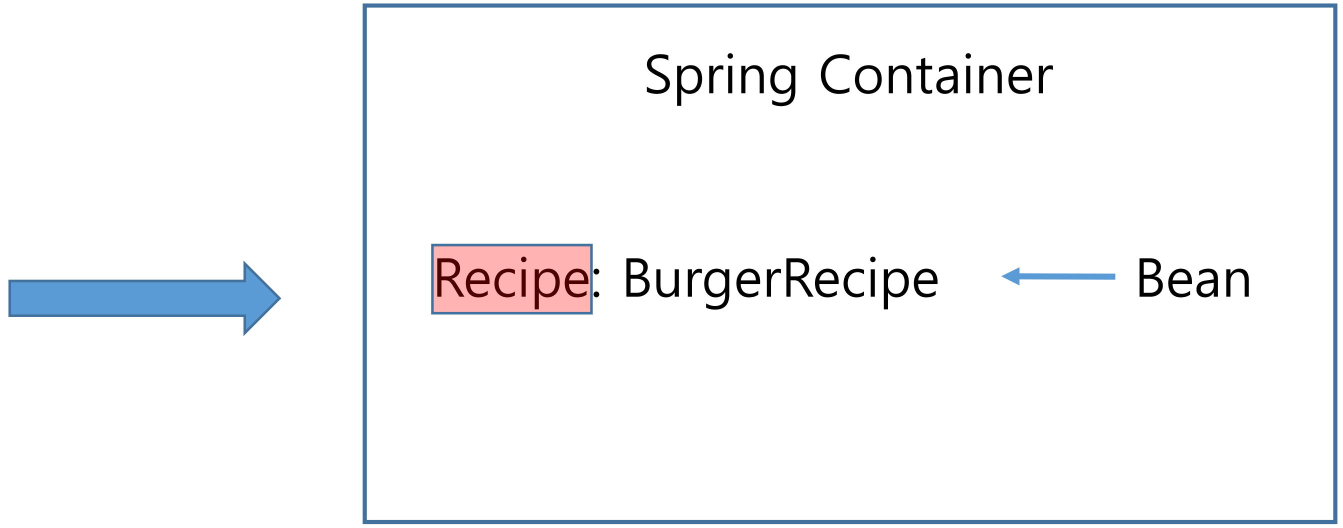
```
class BurgerRecipe {

        List<Step> steps = ... ;

        getSteps() {
                return steps;
        }
}
```

```
class PizzaRecipe {

        List<Step> steps = ... ;

        getSteps() {
                return steps;
        }
}
```

메서드가 매우 많아진다면?

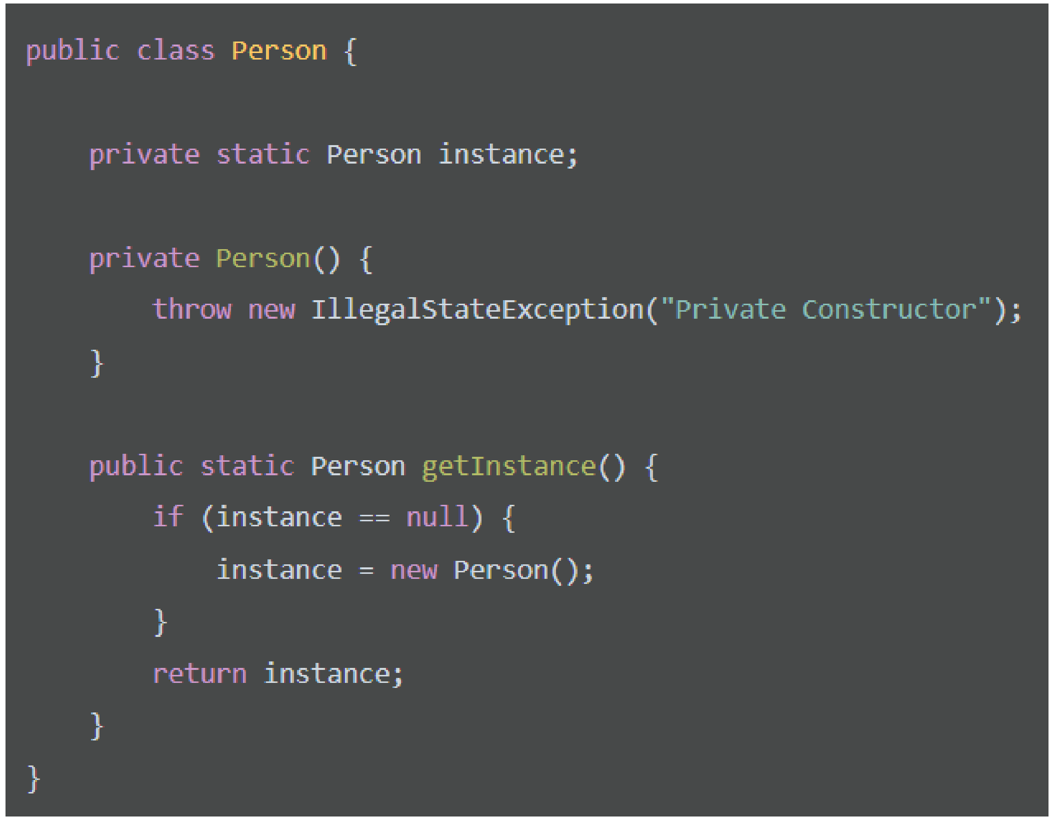
Spring Container
스프링 컨테이너

```
class Chef {

        Recipe recipe = new Recipe();

         cook() {
                recipe.getSteps();
                // 요리 하기
         }

}
```



Singleton
싱글톤

Singleton의 문제



```java
public class Person {

    private static Person instance;

    private Person() {
        throw new IllegalStateException("Private Constructor");
    }

    public static Person getInstance() {
        if (instance == null) {
            instance = new Person();
        }
        return instance;
    }
}
```

```java
public class Person {

    private static Person instance;

    private Person() {
        throw new IllegalStateException("Private Constructor");
    }

    public static Person getInstance() {
        if (instance == null) {
            instance = new Person();
        }
        return instance;
    }
}
```

```java
public class Person {

}
```