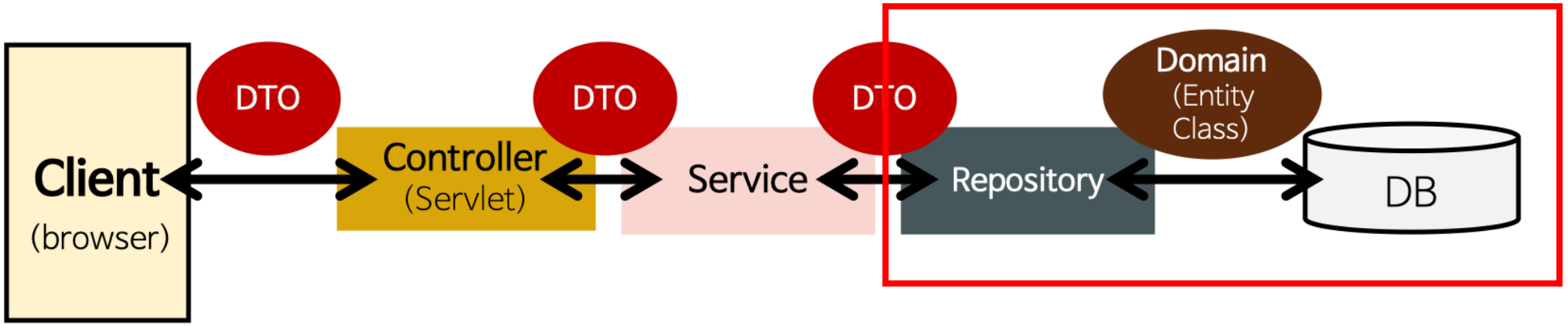
Spring Boot Business Logic

Jae Hyeon Kim

Contents

Review : Spring Boot CRUD



Client **⟨-dto—⟩** controller(web) - service - repository(dao) **⟨—domain(entity)—⟩** DB

Entity

```java
import lombok.*;
import javax.persistence.*;
import java.util.Date;

7개 사용 위치  kiku99
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "tb_post")
@Entity
public class PostEntity {
```

```java
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column
    private String title;

    @Column
    private String content;

    @Column
    private String writer;

    @Column
    private int view_cnt;

    @Column
    private int notice_yn;

    @Column
    private int delete_yn;

    @Column
    private Date created_date;

    @Column
    private Date modified_date;
}
```

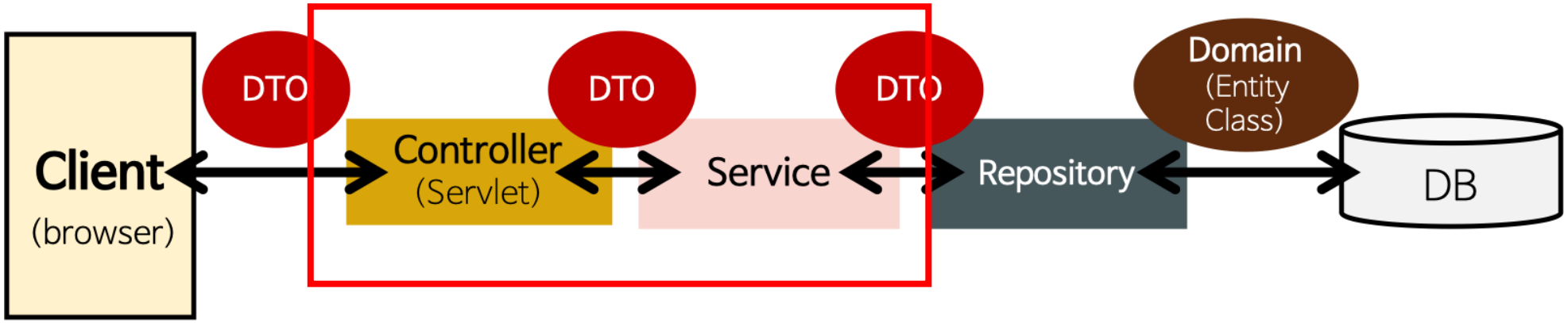
Repository(DAO)

```java
import org.springframework.data.jpa.repository.JpaRepository;

1개 사용 위치  kiku99
public interface PostRepository extends JpaRepository<PostEntity, Integer> {
}
```

Review : Spring Boot CRUD

Client ⟨-dto—⟩ controller(web) - service - repository(dao) ⟨—domain(entity)—⟩ DB

Review : Spring Boot CRUD



Client **〈-dto—〉** controller(web) - service - repository(dao) **〈—domain(entity)—〉** DB

Service & DTO

DTO(Data Transfer Object)

- DTO는 계층 간 데이터 교환 역할을 한다. DB에서 꺼낸 데이터를 저장하는 Entity를 가지고 만드는 일종의 Wrapper라고 볼 수 있는데, Entity를 Controller 같은 클라이언트단과 직접 마주하는 계층에 전달하는 대신 DTO를 사용해 데이터를 교환한다.

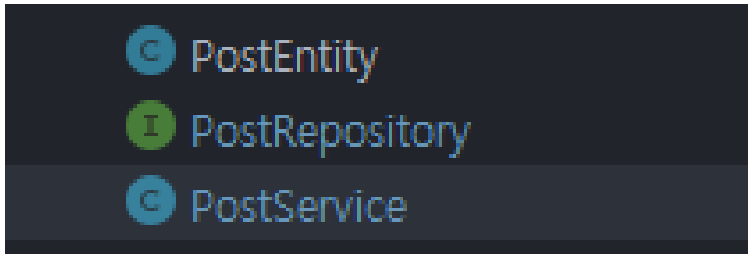
DTO(Data Transfer Object)


```java
7개 사용 위치
@Getter
@NoArgsConstructor
public class PostSaveDto {
    2개 사용 위치
    private String title;
    2개 사용 위치
    private String content;
    2개 사용 위치
    private String writer;
    2개 사용 위치
    private Date created_date;

    1개 사용 위치
    @Builder
    public PostSaveDto(String title, String content, String writer, Date created_date){
        this.title = title;
        this.content = content;
        this.writer = writer;
        this.created_date = created_date;
    }

    1개 사용 위치
    public PostEntity toEntity(){
        return PostEntity.builder()
                .title(title)
                .content(content)
                .writer(writer)
                .created_date(created_date)
                .build();
    }
}
```

Service

- Service의 역할은 DAO가 DB에서 받아온 데이터를 전달받아 가공하고 Controller에게 정보를 보내주는 역할
- 실제 비즈니스 로직을 담고있음

Service



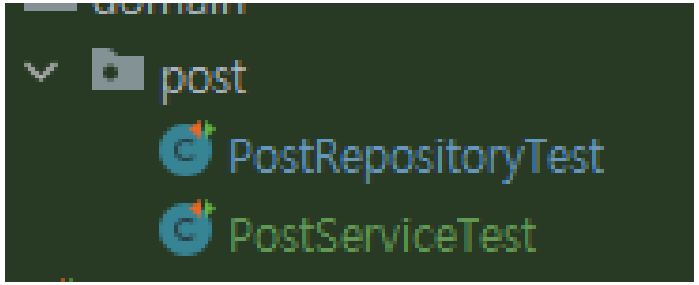

```
2개 사용 위치  kiku99 *
@Service
@RequiredArgsConstructor
public class PostService {

    1개 사용 위치
    private final PostRepository postRepository;

    2개 사용 위치  kiku99 *
    @Transactional
    public Long savePost(final PostSaveDto postSaveDto){
        return postRepository.save(postSaveDto.toEntity()).getId();
    }
}
```

JpaRepository

```java
@NoRepositoryBean
public interface CrudRepository<T, ID> extends Repository<T, ID> {

    Saves a given entity. Use the returned instance for further operations as the save operation might have changed the entity instance completely.
    매개변수: entity – must not be null.
    반환: the saved entity; will never be null.
    던지기: IllegalArgumentException – in case the given entity is null.
            OptimisticLockingFailureException – when the entity uses optimistic locking and has a version attribute with a different value from that found in the persistence store. Also thrown if the entity is assumed to be present but does not exist in the database.
    1개 구현
    <S extends T> S save(S entity);

    Saves all given entities.
    매개변수: entities – must not be null nor must it contain null.
    반환: the saved entities; will never be null. The returned Iterable will have the same size as the Iterable passed as an argument.
    던지기: IllegalArgumentException – in case the given entities or one of its entities is null.
            OptimisticLockingFailureException – when at least one entity uses optimistic locking and has a version attribute with a different value from that found in the persistence store. Also thrown if at least one entity is assumed to be present but does not exist in the database.
    1개 사용 위치  2개 구현
    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    Retrieves an entity by its id.
    매개변수: id – must not be null.
    반환: the entity with the given id or Optional#empty() if none found.
    던지기: IllegalArgumentException – if id is null.
    1개 구현
    Optional<T> findById(ID id);

    Returns whether an entity with the given id exists.
    매개변수: id – must not be null.
    반환: true if an entity with the given id exists, false otherwise.
    던지기: IllegalArgumentException – if id is null.
    1개 구현
    boolean existsById(ID id);
```

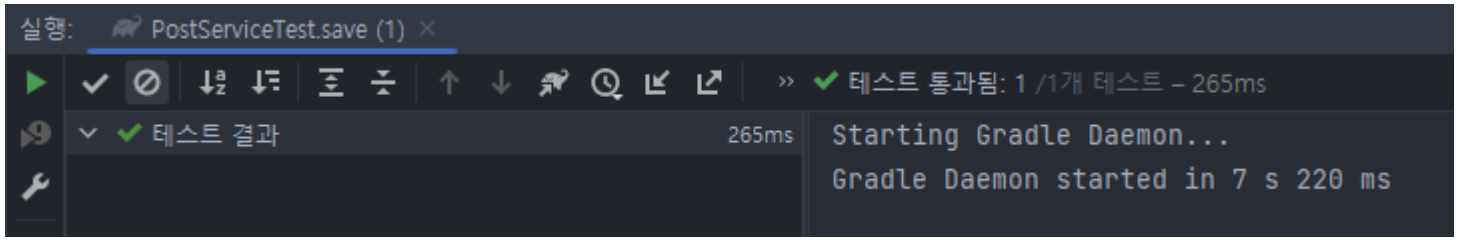
Test Service

Test Service




```java
@SpringBootTest
public class PostServiceTest {

    1개 사용 위치
    @Autowired
    PostService postService;

    @Test
    void save(){
        PostSaveDto postSaveDto = new PostSaveDto( title: "my title",  content: "my content",  writer: "kiku", new Date());
        postService.savePost(postSaveDto);
    }
}
```

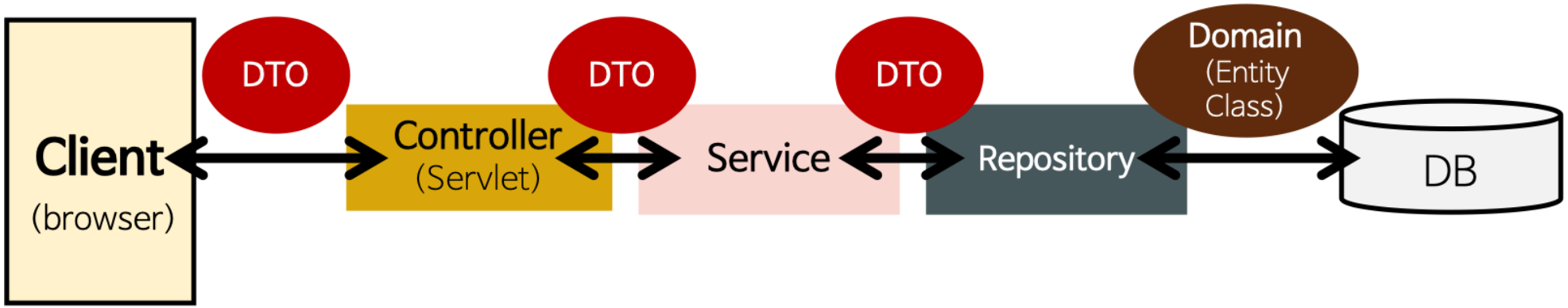
Test Service



WHERE ORDER BY

	id	title	content	writer	view_cnt	notice_yn	delete_yn	created_date	modified_date
1	2	insert test	insert test content	kiku	0	0	0	2022-11-25 15:45:43	<null>
2	3	title	content	writer	0	0	0	2022-12-08 00:08:44	<null>
3	4	my title	my content	kiku	0	0	0	2022-12-09 15:24:15	<null>

Test Service



Client ⟨-dto—⟩ controller(web) - service - repository(dao) ⟨—domain(entity)—⟩ DB