Преобразование DTO в сущность для обновления

Создание и обновление сущности — это похожие операции, но все таки они совпадают не полностью. Различия между операциями приводят к тому, что нам нужно создавать свои DTO под каждую из них. В этом уроке мы научимся реализовывать операцию обновления с учетом DTO.

Возьмем для примера `Post`. Как правило, слаг в постах не меняется после создания. Если его изменить, все ссылки на пост перестанут работать и начнут выдавать ошибку 404. Технически это означает, что для обновления мы должны запретить изменять слаг. Фактически нам придется создать свой DTO для операции обновления.

В итоге мы сделаем три DTO для CRUD всего одной сущности. Может показаться, что это слишком много. К сожалению, здесь нет идеального решения. Свой DTO под каждый тип операции — это самое простое и устойчивое к ошибкам решение, хотя оно и приводит к большему объему кода.

Посмотрим, как выглядит наш DTO для создания сущности:

```
package io.hexlet.spring.dto;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class PostCreateDTO {
    private String slug;
    private String name;
    private String body;
}
```

В таком случае DTO для обновления будет выглядеть так:

```
package io.hexlet.spring.dto;

import lombok.Getter;
```

```
import lombok.Setter;

@Setter
@Getter
public class PostUpdateDTO {
    private String name;
    private String body;
}
```

Второе отличие обновления от создания связано с тем, что мы не создаем новый объект, а меняем уже существующий. То есть правильная реализация обновления выглядит так:

```
var post = postRepository.findById(id);
post.setName(dto.getName());
// остальной код
postRepository.save(post); // UPDATE
```

Учитывая все сказанное выше, мы получим следующий обработчик в контроллере:

```
@RestController
@RequestMapping("/api")
public class PostsController {
    @Autowired
    private PostRepository repository;

    @PutMapping("/posts/{id}")
    @ResponseStatus(HttpStatus.OK)
    public PostDTO update(@RequestBody @Valid PostUpdateDTO postData, @PathVariable Long i
        var post = repository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Not Found"));
        toEntity(postData, post);
        repository.save(post);
        var postDTO = toDTO(post);
        return postDTO;
    }

    private Post toEntity(PostUpdateDTO postDto, Post post) {
        post.setName(postDto.getName());
        post.setBody(postDto.getBody());
        return post;
    }
```

```
    private PostDTO toDTO(Post post) {
        var dto = new PostDTO();
        dto.setId(post.getId());
        dto.setSlug(post.getSlug());
        dto.setName(post.getName());
        dto.setBody(post.getBody());
        dto.setCreatedAt(post.getCreatedAt());
        return dto;
    }
}
```

У этой реализации есть одно важное ограничение — она не поддерживает частичное обновление. Для примера представим, что мы передали ей новое значение и не уточнили, что хотим оставить старое. В таком случае эта реализация просто удалит старое значение. В одном из следующих уроков мы обсудим, как решить эту проблему.

Далее →