PROGRAMMING TECHIE

Build a Full Stack Reddit Clone with – Spring boot and Angular – Part 5

2 Comments

BY SAI UPADHYAYULA | December 14, 2019

Welcome to Part 5 of Build a Full Stack Reddit Clone with Spring boot and Angular series. In Part 4, we saw how to validate JWT using Public Keys and wrote our first secured API – to create and read subreddits.

> The source code of this project, is hosted on Github –
> Backend – https://github.com/SaiUpadhyayula/spring-reddit-clone
> Frontend – https://github.com/SaiUpadhyayula/angular-reddit-clone

What are we building?

In this part, we will write API's to read and create Posts and also we will introduce a new library called **Mapstruct**.

If you are a visual learner like me you can check out the video version of this tutorial below:

What is Mapstruct?

Mapstruct is a code generation library that helps us improve our code by automatically generating the Java Bean Mappings.

What do I mean by that? Take for example our **SubredditService** class, we had two methods that are responsible to map **Subreddit** class to **SubredditDto** class and vice versa.

```
private SubredditDto mapToDto(Subreddit subreddit) {
    return SubredditDto.builder().name(subreddit.getName())
            .id(subreddit.getId())
            .postCount(subreddit.getPosts().size())
            .build();
}

private Subreddit mapToSubreddit(SubredditDto subredditDto) {
    return Subreddit.builder().name("/r/" + subredditDto.getName())
            .description(subredditDto.getDescription())
            .user(authService.getCurrentUser())
            .createdDate(now()).build();
}
```

As you can observe we have to manually build the mapping methods, now imagine if our **Subreddit** is a big entity with more than 5-6 fields. Then it would be difficult and complex to write mapping logic between **Subreddit** and **SubredditDto** manually. By using Mapstruct, we can automatically generate the mapping code at compile time.

Installing Mapstruct

To install Mapstruct, copy the below code inside our **pom.xml** file. You can also read the instructions here

```
..
// In the Properties section
<properties>
    <org.mapstruct.version>1.3.1.Final</org.mapstruct.version>
</properties>
...
// In the dependencies Section
<dependencies>
    <dependency>
        <groupId>org.mapstruct</groupId>
```

```
        <artifactId>mapstruct</artifactId>
        <version>${org.mapstruct.version}</version>
    </dependency>
</dependencies>
...
// In the build Section
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5.1</version> <!-- or newer version -->
            <configuration>
                <source>1.8</source> <!-- depending on your project -->
                <target>1.8</target> <!-- depending on your project -->
                <annotationProcessorPaths>
                    <path>
                        <groupId>org.mapstruct</groupId>
                        <artifactId>mapstruct-processor</artifactId>
                        <version>${org.mapstruct.version}</version>
                    </path>
                     <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>1.18.8</version>
                     </path>
                </annotationProcessorPaths>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Note that we have also added Lombok to the **annotationProcessorPaths** section as the maven compiler plugin should detect both Mapstruct and Lombok, as both these libraries are used to generate code at compile time.

Create Mappings

So now we are ready to use Mapstruct in our project, let's create a class called **SubredditMapper** inside **com.example.springredditclone.mapper** package:

```
package com.example.springredditclone.mapper;

import com.example.springredditclone.dto.SubredditDto;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.Subreddit;
import org.mapstruct.InheritInverseConfiguration;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;

import java.util.List;

@Mapper(componentModel = "spring")
public interface SubredditMapper {

    @Mapping(target = "numberOfPosts", expression = "java(mapPosts(subreddit.getPosts()))")
    SubredditDto mapSubredditToDto(Subreddit subreddit);

    default Integer mapPosts(List<Post> numberOfPosts) {
        return numberOfPosts.size();
    }

    @InheritInverseConfiguration
    @Mapping(target = "posts", ignore = true)
    Subreddit mapDtoToSubreddit(SubredditDto subreddit);
}
```

Let's go through what we created in this class:

- First, we annotated the **SubredditMapper** with **@Mapper(componentModel='spring')** annotation to specify that this interface is a Mapstruct Mapper and Spring should identify it as a component and should be able to inject it into other components like **SubredditService**.
- The first method inside **SubredditMapper** class **mapSubredditToDto()**, contains only one **@Mapping** annotation for the target field **numberOfPosts**, in this case, we are mapping from List<Posts> to an Integer, this kind of mapping is not straight forward and we need to write our logic. We can do that by using the **expression** field and pass the method definition for **mapPosts()** which returns an Integer.
- Notice that for all remaining fields, the **@Mapping** annotation is not needed as they are implicitly added at compile time.
- We can create reverse mappings from **SubredditDto** to **Subreddit** by annotating a method with **InheritInverseConfiguration**. This annotation reverse's the mapping which exists to convert from **Subreddit** to **SubredditDto**

If you compile the above interface, we can see the generated class under the folder **target/generated-sources/annotations/com.example.springredditclone.mapper**

This is how the generated class implementation looks like:

```
package com.example.springredditclone.mapper;

import com.example.springredditclone.dto.SubredditDto;
import com.example.springredditclone.dto.SubredditDto.SubredditDtoBuilder;
import com.example.springredditclone.model.Subreddit;
import com.example.springredditclone.model.Subreddit.SubredditBuilder;
import javax.annotation.Generated;
import org.springframework.stereotype.Component;

@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2019-12-12T05:34:58+0100",
    comments = "version: 1.3.1.Final, compiler: javac, environment: Java 1.8.0_191 (Oracle Corporation)"
)
@Component
public class SubredditMapperImpl implements SubredditMapper {

    @Override
    public SubredditDto mapSubredditToDto(Subreddit subreddit) {
        if ( subreddit == null ) {
            return null;
        }

        SubredditDtoBuilder subredditDto = SubredditDto.builder();

        subredditDto.id( subreddit.getId() );
        subredditDto.name( subreddit.getName() );
        subredditDto.description( subreddit.getDescription() );
```

```
            subredditDto.numberOfPosts( mapPosts(subreddit.getPosts()) );

            return subredditDto.build();
        }

        @Override
        public Subreddit mapDtoToSubreddit(SubredditDto subreddit) {
            if ( subreddit == null ) {
                return null;
            }

            SubredditBuilder subreddit1 = Subreddit.builder();

            subreddit1.id( subreddit.getId() );
            subreddit1.name( subreddit.getName() );
            subreddit1.description( subreddit.getDescription() );

            return subreddit1.build();
        }
    }
```

Use Mappers inside Spring Components

Now let's inject **SubredditMapper** into **SubredditService** class and use it in the required places, the refactored code for **SubredditService** looks like below:

SubredditService.java

```
package com.example.springredditclone.service;

import com.example.springredditclone.dto.SubredditDto;
import com.example.springredditclone.exceptions.SpringRedditException;
import com.example.springredditclone.mapper.SubredditMapper;
import com.example.springredditclone.model.Subreddit;
import com.example.springredditclone.repository.SubredditRepository;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

import static java.util.stream.Collectors.toList;

@Service
@AllArgsConstructor
@Slf4j
public class SubredditService {

    private final SubredditRepository subredditRepository;
    private final SubredditMapper subredditMapper;

    @Transactional
    public SubredditDto save(SubredditDto subredditDto) {
        Subreddit save = subredditRepository.save(subredditMapper.mapDtoToSubreddit(subredditDto));
        subredditDto.setId(save.getId());
        return subredditDto;
    }

    @Transactional(readOnly = true)
    public List<SubredditDto> getAll() {
        return subredditRepository.findAll()
                .stream()
                .map(subredditMapper::mapSubredditToDto)
                .collect(toList());
    }
```

```
    public SubredditDto getSubreddit(Long id) {
        Subreddit subreddit = subredditRepository.findById(id)
                .orElseThrow(() -> new SpringRedditException("No subreddit found with ID - " + id));
        return subredditMapper.mapSubredditToDto(subreddit);
    }
}
```

Implement API's to Create and Read Posts

So the next step is to create APIs to create Posts inside the Subreddits and Read those posts. First let's create the controller class – **PostController** inside **com.example.springredditclone.controller** package.

PostController.java

```
package com.example.springredditclone.controller;

import com.example.springredditclone.dto.PostRequest;
import com.example.springredditclone.dto.PostResponse;
import com.example.springredditclone.service.PostService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

import static org.springframework.http.ResponseEntity.status;

@RestController
@RequestMapping("/api/posts/")
@AllArgsConstructor
public class PostController {

    private final PostService postService;

    @PostMapping
    public ResponseEntity<Void> createPost(@RequestBody PostRequest postRequest) {
        postService.save(postRequest);
        return new ResponseEntity<>(HttpStatus.CREATED);
    }

    @GetMapping
    public ResponseEntity<List<PostResponse>> getAllPosts() {
        return status(HttpStatus.OK).body(postService.getAllPosts());
    }

    @GetMapping("/{id}")
    public ResponseEntity<PostResponse> getPost(@PathVariable Long id) {
        return status(HttpStatus.OK).body(postService.getPost(id));
    }

    @GetMapping("by-subreddit/{id}")
    public ResponseEntity<List<PostResponse>> getPostsBySubreddit(Long id) {
        return status(HttpStatus.OK).body(postService.getPostsBySubreddit(id));
    }

    @GetMapping("by-user/{name}")
    public ResponseEntity<List<PostResponse>> getPostsByUsername(String username) {
        return status(HttpStatus.OK).body(postService.getPostsByUsername(username));
    }
}
```

You can refer to the below table to see the different endpoints we created, they should be pretty self-explanatory.

Mapping	HTTP Method	Method Name
/api/posts	POST	createPost
/api/posts/	GET	getAllPosts
/api/posts/{id}	GET	getPost
/api/posts/by-subreddit/{id}	GET	getPostsBySubreddit
/api/posts/by-user/{name}	GET	getPostsByUsername

We are using **PostRequest** class as DTO, to receive Post data from client.

PostRequest.java

```
package com.example.springredditclone.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class PostRequest {
    private Long postId;
    private String subredditName;
    private String postName;
    private String url;
    private String description;
}
```

Alright, now let's implement all the required methods inside **PostService** class:

```
package com.example.springredditclone.service;

import com.example.springredditclone.dto.PostRequest;
import com.example.springredditclone.dto.PostResponse;
import com.example.springredditclone.exceptions.PostNotFoundException;
import com.example.springredditclone.exceptions.SubredditNotFoundException;
import com.example.springredditclone.mapper.PostMapper;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.Subreddit;
import com.example.springredditclone.model.User;
import com.example.springredditclone.repository.PostRepository;
import com.example.springredditclone.repository.SubredditRepository;
import com.example.springredditclone.repository.UserRepository;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

import static java.util.stream.Collectors.toList;

@Service
@AllArgsConstructor
@Slf4j
@Transactional
```

```
public class PostService {

    private final PostRepository postRepository;
    private final SubredditRepository subredditRepository;
    private final UserRepository userRepository;
    private final AuthService authService;
    private final PostMapper postMapper;

    @Transactional(readOnly = true)
    public PostResponse getPost(Long id) {
        Post post = postRepository.findById(id)
                .orElseThrow(() -> new PostNotFoundException(id.toString()));
        return postMapper.mapToDto(post);
    }

    @Transactional(readOnly = true)
    public List<PostResponse> getAllPosts() {
        return postRepository.findAll()
                .stream()
                .map(postMapper::mapToDto)
                .collect(toList());
    }

    public void save(PostRequest postRequest) {
        Subreddit subreddit = subredditRepository.findByName(postRequest.getSubredditName())
                .orElseThrow(() -> new SubredditNotFoundException(postRequest.getSubredditName()));
        postRepository.save(postMapper.map(postRequest, subreddit, authService.getCurrentUser()));
    }

    @Transactional(readOnly = true)
    public List<PostResponse> getPostsBySubreddit(Long subredditId) {
        Subreddit subreddit = subredditRepository.findById(subredditId)
                .orElseThrow(() -> new SubredditNotFoundException(subredditId.toString()));
        List<Post> posts = postRepository.findAllBySubreddit(subreddit);
        return posts.stream().map(postMapper::mapToDto).collect(toList());
    }

    @Transactional(readOnly = true)
    public List<PostResponse> getPostsByUsername(String username) {
        User user = userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException(username));
        return postRepository.findByUser(user)
                .stream()
                .map(postMapper::mapToDto)
                .collect(toList());
    }
}
```

PostMapper.java

```
package com.example.springredditclone.mapper;

import com.example.springredditclone.dto.PostRequest;
import com.example.springredditclone.dto.PostResponse;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.Subreddit;
import com.example.springredditclone.model.User;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;

@Mapper(componentModel = "spring")
public interface PostMapper {
    @Mapping(target = "createdDate", expression = "java(java.time.Instant.now())")
    @Mapping(target = "subreddit", source = "subreddit")
    @Mapping(target = "user", source = "user")
    @Mapping(target = "description", source = "postRequest.description")
```

```
    Post map(PostRequest postRequest, Subreddit subreddit, User user);

    @Mapping(target = "id", source = "postId")
    @Mapping(target = "postName", source = "postName")
    @Mapping(target = "description", source = "description")
    @Mapping(target = "url", source = "url")
    @Mapping(target = "subredditName", source = "subreddit.name")
    @Mapping(target = "userName", source = "user.username")
    PostResponse mapToDto(Post post);
}
```

PostResponse.java

```
package com.example.springredditclone.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class PostResponse {
    private Long id;
    private String postName;
    private String url;
    private String description;
    private String userName;
    private String subredditName;
}
```

- The **save()** method is first mapping the **PostRequest** object to our **Post** entity using the **PostMapper.map()** method, observe that in our **Post** entity we need to fill some fields like **subreddit**, **createdDate**, **user** which are not present in the **PostRequest** object.
- So we will first query the necessary information and pass them to the **PostMapper.map** method.
- For fields **createdDate** and **description** we are using Mapstruct expressions to calculate the source value.
- Next, we have all the methods to retrieve posts – based on a subreddit, user and post id.
- Here we are using the **PostMapper.mapToDto()** to generate a **PostResponse** object.

Testing time

So now let's start our application, and see if we are able to successfully create our first Post.

Open **Postman** and make a POST call to **http://localhost:8080/api/posts** with the body:

```
{
     "subredditName":"First Subreddit",
     "postName":"This is my first post",
     "url":"http://google.com",
     "description":"First post"
 }
```

You should receive a **201 OK** response.

Now make a GET call to **http://localhost:8080/api/posts** you should see the Post details.

What's next?

Alright, so we saw what is Mapstruct and how to use this library to generate mapping logic in our application, we also create API to Create and Read Posts.

In the next article, we will see how to create more API's to submit votes and comments on our Posts.

I hope these articles are helpful to you, see you in the next article.

Until then, happy coding.

About the author

Sai Upadhyayula

Adil

January 10, 2020 at 6:22 pm

Subscribe now to get the latest updates!

Name Email Sign Up

Hi

Thank you. I will try to post as soon as I get some time. I will also post the Angular part of this tutorial.

Comments are closed.