PROGRAMMING TECHIE

Build a Full Stack Reddit Clone with – Spring boot and Angular – Part 6

0 Comments

BY SAI UPADHYAYULA | February 18, 2020

Welcome to Part 6 of Build a Full Stack Reddit Clone with Spring boot and Angular series. In Part 5, we saw how to create an API to create Posts in our application. We also saw how to write mapping logic in our project using the library Mapstruct.

What are we building?

In this part, we will write APIs to comment on the Posts created by a user. We will also send Comment Notification emails to the creator of the Post.

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

> 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

Implementing API for Managing Comments

We will start creating APIs for posting Comments and Votes on the posts created by the user.

In the below table, you can find the details about each mapping in our REST API for Comments ie. **CommentsController.java** class.

Mapping	HTTP Method	Method Name
/api/comments/	POST	createComments
/api/comments/by-postId/{postId}	GET	getAllCommentsForPost
/api/comments/by-user/{userName}	GET	getCommentsByUsername

CommentsController.java

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

import com.example.springredditclone.dto.CommentsDto;
import com.example.springredditclone.service.CommentService;
import lombok.AllArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

import static org.springframework.http.HttpStatus.CREATED;
import static org.springframework.http.HttpStatus.OK;
import static org.springframework.http.ResponseEntity.status;

@RestController
@RequestMapping("/api/comments/")
@AllArgsConstructor
public class CommentsController {

    private final CommentService commentService;
```

```
    @PostMapping
    public ResponseEntity<Void> createComment(@RequestBody CommentsDto commentsDto) {
        commentService.createComment(commentsDto);
        return new ResponseEntity<>(CREATED);
    }

    @GetMapping
    public ResponseEntity<List<CommentsDto>> getAllCommentsForPost(@RequestParam("postId") Long postId) {
        return status(OK)
                .body(commentService.getCommentByPost(postId));
    }

    @GetMapping
    public ResponseEntity<List<CommentsDto>> getAllCommentsByUser(@RequestParam("userName") String userName) {
        return status(OK).body(commentService.getCommentsByUser(userName));
    }
}
```

CommentsDto.java

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

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

import java.time.Instant;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CommentsDto {
    private Long id;
    private Long postId;
    private Instant createdDate;
    private String text;
    private String userName;
}
```

- First off, we have our **CommentsController** class where we have 3 endpoints which we mentioned previously in a table.
- The **createComment** takes an object of type **CommentsDtos** as input, the creation logic is handled inside the CommentService class
- Once the comment is created we are returning a 201 response back to the client (CREATED).
- After that we have the **getAllCommentsForPost()** and **getAllCommentsByUser()** methods where we are retrieving the relevant comments for given user/post from the **CommentService**.
- These methods are returning and OK which means HTTP Status 200.

CommentService.java

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

import com.example.springredditclone.dto.CommentsDto;
import com.example.springredditclone.exceptions.PostNotFoundException;
import com.example.springredditclone.mapper.CommentMapper;
import com.example.springredditclone.model.Comment;
import com.example.springredditclone.model.NotificationEmail;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.User;
```

```
import com.example.springredditclone.repository.CommentRepository;
import com.example.springredditclone.repository.PostRepository;
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 CommentService {

    //TODO: Construct POST URL
    private static final String POST_URL = "";

    private final CommentMapper commentMapper;
    private final PostRepository postRepository;
    private final CommentRepository commentRepository;
    private final UserRepository userRepository;
    private final AuthService authService;
    private final MailContentBuilder mailContentBuilder;
    private final MailService mailService;

    public void createComment(CommentsDto commentsDto) {
        Post post = postRepository.findById(commentsDto.getPostId())
                .orElseThrow(() -> new PostNotFoundException(commentsDto.getPostId().toString()));
        Comment comment = commentMapper.map(commentsDto, post, authService.getCurrentUser());
        commentRepository.save(comment);

        String message = mailContentBuilder.build(post.getUser().getUsername() + " posted a comment on your
post." + POST_URL);
        sendCommentNotification(message, post.getUser());
    }

    public List<CommentsDto> getCommentByPost(Long postId) {
        Post post = postRepository.findById(postId)
                .orElseThrow(() -> new PostNotFoundException(postId.toString()));
        return commentRepository.findByPost(post)
                .stream()
                .map(commentMapper::mapToDto)
                .collect(toList());
    }

    public List<CommentsDto> getCommentsByUser(String userName) {
        User user = userRepository.findByUsername(userName)
                .orElseThrow(() -> new UsernameNotFoundException(userName));
        return commentRepository.findAllByUser(user)
                .stream()
                .map(commentMapper::mapToDto)
                .collect(toList());
    }

    private void sendCommentNotification(String message, User user) {
        mailService.sendMail(new NotificationEmail(user.getUsername() + " Commented on your post",
user.getEmail(), message));
    }
}
```

CommentRepository.java

```
package com.example.springredditclone.repository;

import com.example.springredditclone.model.Comment;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CommentRepository extends JpaRepository<Comment, Long> {
    List<Comment> findByPost(Post post);

    List<Comment> findAllByUser(User user);
}
```

CommentMapper.java

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

import com.example.springredditclone.dto.CommentsDto;
import com.example.springredditclone.model.Comment;
import com.example.springredditclone.model.Post;
import com.example.springredditclone.model.User;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;

@Mapper(componentModel = "spring")
public interface CommentMapper {
    @Mapping(target = "id", ignore = true)
    @Mapping(target = "text", source = "commentsDto.text")
    @Mapping(target = "createdDate", expression = "java(java.time.Instant.now())")
    @Mapping(target = "post", source = "post")
    Comment map(CommentsDto commentsDto, Post post, User user);

    @Mapping(target = "postId", expression = "java(comment.getPost().getPostId())")
    @Mapping(target = "userName", expression = "java(comment.getUser().getUsername())")
    CommentsDto mapToDto(Comment comment);
}
```

- Inside the **CommentService** class, we have our **createComment** method where we are creating the comment, and right after that we are sending a notification email to the user through **sendCommentNotification()**
- We are using the **CommentMapper** interface to map our **Comment** and **CommentsDto** class.
- Notice that when using **Instant.now()** inside the mapping for the field **createDate** we are using the fully qualified class name of Instant class. This is because as this expression is inside a String, Mapstruct is not able to recognize and import the relevant import statements for this class.

Testing

Now its time to test our implementation, let's start up the server and open Postman. We will first login to our application to get the **Bearer Token**.

Login to the application

Now pass the Bearer Token inside the **Authorization** tab, first by selecting **Bearer Token** and pasting it in the **Token** input field. After that add a request body which contains **postId**, **text** and **userName**

Create Comment

Create Comment with Request Body

Now let's check whether we received an email notification for the comment or not. Login to your MailTrap.io account and check your inbox. You should see an email that looks like below.

Comment Notification

Oops did you spot the bug ?? Our email message is looking weird because we are rendering an html inside the **span** tag, this is a small bug which can be fixed by replacing the code inside the **MailService**.

MailService.java

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

import com.example.springredditclone.exceptions.SpringRedditException;
import com.example.springredditclone.model.NotificationEmail;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.mail.MailException;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.mail.javamail.MimeMessagePreparator;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

@Service
@AllArgsConstructor
```

```
@Slf4j
class MailService {

    private final JavaMailSender mailSender;
    private final MailContentBuilder mailContentBuilder;

    @Async
    void sendMail(NotificationEmail notificationEmail) {
        MimeMessagePreparator messagePreparator = mimeMessage -> {
            MimeMessageHelper messageHelper = new MimeMessageHelper(mimeMessage);
            messageHelper.setFrom("springreddit@email.com");
            messageHelper.setTo(notificationEmail.getRecipient());
            messageHelper.setSubject(notificationEmail.getSubject());
            //Before messageHelper.setText(mailContentBuilder.build(notificationEmail.getBody()));
// After
messageHelper.setText(notificationEmail.getBody());
        };
        try {
            mailSender.send(messagePreparator);
            log.info("Activation email sent!!");
        } catch (MailException e) {
            throw new SpringRedditException("Exception occurred when sending mail to " +
notificationEmail.getRecipient());
        }
    }

}
```

Now create another Comment and this time our email looks like below:

Now let's call the GET endpoints to check if they are working as expected or not.

GET – /api/comments/by-post/{postId}

Conclusion

So that is it for this tutorial, we are slowly getting to the end of the backend part of this Reddit Clone with Spring Boot and Angular Series. In the next tutorial, we will see how to cast votes for the posts created by the user.

We will create a similar API like above and will also do some refactoring of the existing code.

I hope you are finding this tutorial series helpful, I will see you in the next part, until then H

About the author
Sai Upadhyayula