PROGRAMMING TECHIE

Build a Full Stack Reddit Clone with – Spring boot and Angular – Part 16

0 Comments

BY SAI UPADHYAYULA | April 20, 2020

Reddit clone with Spring boot and Angular
#16
Post Comments & Implement User Profile Page

Share Tweet Pin

Welcome to Full Stack Reddit Clone with Spring boot and Angular – Part 16. In Part 15, we create Subreddits and Posts in our Angular Application.

In this article we are going to:

- Implement Page to View Posts
- Implement User Profile Page to check Posts and Comments submitted by the user.

If you are a visual learner like me you can check out the below Youtube Tutorial

Thank you for visiting. You can now buy me a coffee!

https://www.youtube.com/watch?v=OLT3fNcHQh4

Let's start coding.

Download Source Code

> **Source code for Angular application** – https://github.com/SaiUpadhyayula/angular-reddit-clone
> **Source code for Spring boot backend** –https://github.com/SaiUpadhyayula/spring-reddit-clone

Develop Page to View Posts

So in the last article, we developed the logic to create Posts, but we still don't have a way to drill-down and read the post. Let's implement that component, and later we will also add the functionality to post comments to this component.

Let's generate a component called as **view-post** under the **post** folder. You can find the command to generate the component in the below image.

We want to display this component when the user clicks on Read Post button on the home page. So first let's add a click event for the **Read Post** button in `post-tile.component.html`

`post-tile.component.html`

```
<!-- Display Posts-->
<div class="row post" *ngFor="let post of posts$">
  <app-vote-button [post]="post"></app-vote-button>
  <!-- Section to Display Post Information-->
```

```
      <div class="col-md-11">
        <span class="subreddit-info">
          <span class="subreddit-text"><a class="posturl" routerLink="">{{post.subredditName}}</a></span>
          <span> . Posted by <a class="username" routerLink="/user/{{post.userName}}">{{post.userName}}</a>
    </span>
          <span> . {{post.duration}}</span>
        </span>
        <hr />
        <div class="post-title">
          <a class="postname" href="{{post.url}}">{{post.postName}}</a>
        </div>
        <div>
          <p class="post-text" [innerHtml]="post.description"></p>
        </div>
        <hr />
        <span>
          <a class="btnCommments" role="button">
            <fa-icon [icon]="faComments"></fa-icon>
            Comments({{post.commentCount}})
          </a>
          <button class="login" (click)="goToPost(post.id)">
            Read Post
          </button>
        </span>
      </div>
    </div>
```

post-tile.component.ts

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { PostService } from '../post.service';
import { PostModel } from '../post-model';
import { faComments } from '@fortawesome/free-solid-svg-icons';
import { Router } from '@angular/router';

@Component({
  selector: 'app-post-tile',
  templateUrl: './post-tile.component.html',
  styleUrls: ['./post-tile.component.css'],
  encapsulation: ViewEncapsulation.None,
})
export class PostTileComponent implements OnInit {

  posts$: Array<PostModel>;
  faComments = faComments;

  constructor(private postService: PostService, private router: Router) {
    this.postService.getAllPosts().subscribe(post => {
      this.posts$ = post;
    });
  }

  ngOnInit(): void {
  }

  goToPost(id: number): void {
    this.router.navigateByUrl('/view-post/' + id);
  }
}
```

- So if you observe in the post-tile.component.html file, we added a click directive anc with input as `post.id`

- Inside the `post-tile.component.ts` we first injected the Router class from `@angular/router` and added the method `goToPost()` where we are navigating to the URL **'/view-post/:id"**

Now to make this work, we have to define the route to **view-post** inside the `app-routing.module.ts` file.

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { SignupComponent } from './auth/signup/signup.component';
import { LoginComponent } from './auth/login/login.component';
import { HomeComponent } from './home/home.component';
import { CreatePostComponent } from './post/create-post/create-post.component';
import { CreateSubredditComponent } from './subreddit/create-subreddit/create-subreddit.component';
import { ListSubredditsComponent } from './subreddit/list-subreddits/list-subreddits.component';
import { ViewPostComponent } from './post/view-post/view-post.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'view-post/:id', component: ViewPostComponent },
  { path: 'list-subreddits', component: ListSubredditsComponent },
  { path: 'create-post', component: CreatePostComponent },
  { path: 'create-subreddit', component: CreateSubredditComponent },
  { path: 'sign-up', component: SignupComponent },
  { path: 'login', component: LoginComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

So we added an entry inside the `routes` array, where we are mapping the path `view-post/:id` to the `ViewPostComponent`. Where id is the query parameter, the id of the post.

Now copy the below HTML and CSS code into the `view-post.component.html` and `view-post.component.css` files.

view-post.component.html

```
<div class="container">
  <div class="row">
    <hr />
    <div class="col-md-9">
      <div class="row post">
        <div class="col-md-1">
          <app-vote-button [post]="post"></app-vote-button>
        </div>
        <div class="col-md-11">
          <span>
            <span class="subreddit-text"><a class="post-url" href="">{{post.subredditName}}</a></span>
            <span> . Posted
                <span> {{post.duration}} </span>
                by
              <a *ngIf="post.userName === null" class="username" href="">Anonymous</a>
              <a *ngIf="post.userName != null" class="username" href="">{{post.userName}}</a>
            </span>
          </span>
          <hr />
          <a routerLink="post.url" class="post-title">{{post.postName}}</a>
          <div>
            <p class="post-text" [innerHtml]="post.description"></p>
          </div>
        </div>
      </div>
    </div>
    <div class="col-md-3">
      <sidebar></sidebar>
      <sidebar-view-subreddit></sidebar-view-subreddit>
    </div>
  </div>
</div>
```

view-post.component.ts

```
import { Component, OnInit } from '@angular/core';
import { PostService } from 'src/app/shared/post.service';
import { ActivatedRoute } from '@angular/router';
import { PostModel } from 'src/app/shared/post-model';
import { throwError } from 'rxjs';

@Component({
  selector: 'app-view-post',
  templateUrl: './view-post.component.html',
  styleUrls: ['./view-post.component.css']
})
export class ViewPostComponent implements OnInit {

  postId: number;
  post: PostModel;

  constructor(private postService: PostService, private activateRoute: ActivatedRoute) {
    this.postId = this.activateRoute.snapshot.params.id;
    this.postService.getPost(this.postId).subscribe(data => {
      this.post = data;
    }, error => {
      throwError(error);
    });
  }

  ngOnInit(): void {
  }

}
```

- In the HTML file, we are first displaying the Vote button Component using the `app-vote-button` selector, and passing the `post` object as input to the component.
- Inside the `ViewPostComponent`, we are first injecting the `PostService` and `ActivatedRouter` classes, and inside the constructor, we access the incoming query-param id and assign it to the `postId` variable.
- After that, we are reading the post from the `PostService` using the `postService.getPost()` method, which returns an `Observable<PostModel>`
- We subscribe to the response and assign the response, to the **post** variable, if there is an error, then we throw an error using the `throwError` method.

If you open the application, and click on the Read Post button on any of the post, you will see the below screen.

Functionality to Add Comments

In any kind of forum website like Reddit, Stackoverflow the functionality to add Comments is very important. Let's go ahead and implement this functionality also in our application.

Let's update our view-post.component.html file with the below code, we are basically updating the page with a form to submit comments.

`view-post.component.html`

```
<div class="container">
  <div class="row">
    <hr />
    <div class="col-md-9">
      <div class="row post">
        <div class="col-md-1">
          <app-vote-button [post]="post"></app-vote-button>
        </div>
        <div class="col-md-11">
          <span>
            <span class="subreddit-text"><a class="post-url" href="">{{post.subredditName}}</a></span>
            <span> . Posted
              <span> {{post.duration}} </span>
              by
              <a *ngIf="post.userName === null" class="username" href="">Anonymous</a>
              <a *ngIf="post.userName != null" class="username" href="">{{post.userName}}</a>
            </span>
          </span>
          <hr />
          <a routerLink="post.url" class="post-title">{{post.postName}}</a>
          <div>
            <p class="post-text" [innerHtml]="post.description"></p>
          </div>
          <div class="post-comment">
            <form [formGroup]="commentForm" (ngSubmit)="postComment()">
              <div class="form-group">
                <textarea class="form-control" [formControlName]="'text'" placeholder="Your Thoughts?">
</textarea>
              </div>
              <button type="submit" class="login float-right">Comment</button>
            </form>
          </div>
          <div style="margin-top: 60px;" *ngFor="let comment of comments">
            <div class="comment">
              <div class="username">
                <a routerLink="/user/comment.username">{{comment.userName}}</a>
              </div>
              <div>
                <p>{{comment.duration}}</p>
              </div>
              <b>{{comment.text}}</b>
            </div>
```

```
              <hr />
            </div>
          </div>
        </div>
      </div>
      <div class="col-md-3">
        <sidebar></sidebar>
        <sidebar-view-subreddit></sidebar-view-subreddit>
      </div>
    </div>
  </div>
```

view-post.component.css

```
.post-title {
  font-size: 28px;
  font-weight: bold;
  opacity: 1;
}

.post-title:hover {
  opacity: 0.6;
}

.subreddit-text {
  font-weight: bold;
}

.post-url {
  color: black;
}

.username{
    color: gray;
}

.post-text{
    margin-top: 10px;
}

.post{
    --post-line-color: #ccc;
    border: 1px solid #ccc;
    margin-top: 10px;
    margin-bottom: 10px;
    overflow: hidden;
    background-color: rgba(255,255,255,0.8);
    color: #878A8C;
    position: relative;
    border-radius: 4px;
    padding:5px;
}

.comment{
    --post-line-color: #ccc;
    border: 1px solid #ccc;
    margin-bottom: 10px;
    overflow: hidden;
    background-color: rgba(255,255,255,0.8);
    color: #878A8C;
    position: relative;
    border-radius: 4px;
    padding:5px;
}
```

Before implementing the functionality to submit comments, as always, let's refer to the Swagger REST API documentation at **http://localhost:8080/swagger-ui.html**

We need to create a class which acts as the request payload for Create Comments API, I am going to create a file named `comment-payload.ts` under a new folder named **comments**

comment-payload.ts

```
export class CommentPayload{
    text: string;
    postId: string;
    username?:string;
    duration?: string;
}
```

Before implementing the logic, we need to create a service class to make POST calls to the Comments API. Open a terminal and type the following command.

Now let's go ahead and implement the method **submit and display** comments in `view-post.component.ts` file

view-component.ts

```
import { Component, OnInit } from '@angular/core';
import { PostService } from 'src/app/shared/post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { PostModel } from 'src/app/shared/post-model';
import { throwError } from 'rxjs';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { CommentPayload } from 'src/app/comment/comment.payload';
import { CommentService } from 'src/app/comment/comment.service';

@Component({
  selector: 'app-view-post',
  templateUrl: './view-post.component.html',
  styleUrls: ['./view-post.component.css']
})
export class ViewPostComponent implements OnInit {

  postId: number;
  post: PostModel;
  commentForm: FormGroup;
  commentPayload: CommentPayload;
  comments: CommentPayload[];

  constructor(private postService: PostService, private activateRoute: ActivatedRoute,
    private commentService: CommentService, private router: Router) {
    this.postId = this.activateRoute.snapshot.params.id;

    this.commentForm = new FormGroup({
      text: new FormControl('', Validators.required)
    });
    this.commentPayload = {
      text: '',
      postId: this.postId
    };
  }

  ngOnInit(): void {
    this.getPostById();
    this.getCommentsForPost();
  }

  postComment() {
    this.commentPayload.text = this.commentForm.get('text').value;
    this.commentService.postComment(this.commentPayload).subscribe(data => {
      this.commentForm.get('text').setValue('');
      this.getCommentsForPost();
    }, error => {
      throwError(error);
    })
  }

  private getPostById() {
    this.postService.getPost(this.postId).subscribe(data => {
      this.post = data;
    }, error => {
      throwError(error);
    });
  }

  private getCommentsForPost() {
    this.commentService.getAllCommentsForPost(this.postId).subscribe(data => {
      this.comments = data;
    }, error => {
      throwError(error);
    });
  }

}
```

Submitting the comments

- We first declared the variable commentForm of type FormGroup and initialized it inside the constructor. There is a FormControl assigned to the FormGroup, which is initialized to an empty value and we have also defined a Validator, which makes sure that the given value is not empty.
- Next, we declared and initialized the CommentPayload object, we will use this when making a POST call to the Comment API.
- We declared the method postComment(), which reads the value for FormControl variable – text from the FormGroup – commentForm. We then assign the value to the text field of the CommentPayload object.
- Then we are calling the postComment() method inside the CommentService, which returns an Observable, so we subscribe to it and when we receive a success response we are resetting the text variable to an empty value.

Displaying Comments

After submitting the comments, we also implemented the logic to Display Comments:

- We are calling the method `getCommentsForPost()` inside the constructor where we are calling the `getAllCommentsForPost()` from `CommentService`.
- As this returns an Observable we are subscribing to it and assigning the response to the `comments` variable.
- We also have the method `getPostById()` which are reading the post information and assigning it to the post variable

comment-service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CommentPayload } from './comment.payload';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class CommentService {

  constructor(private httpClient: HttpClient) { }

  getAllCommentsForPost(postId: number): Observable<CommentPayload[]> {
    return this.httpClient.get<CommentPayload[]>('http://localhost:8080/api/comments/by-post/' + postId);
  }

  postComment(commentPayload: CommentPayload): Observable<any> {
    return this.httpClient.post<any>('http://localhost:8080/api/comments/', commentPayload);
  }
}
```

Fixing a Bug

Before we go ahead and test our implementation, we have a bug in our `token-interceptor.ts` file, this can be identified only in the case of an expired Authentication Token. If you are following along with this tutorial from the beginning, chances are its not so hard to replicate this bug.

In fact, we have 2 bugs, let's have a look at them:

- The first one – Inside the `intercept()` method, we are retrieving the token from the LocalStorage and cloning the request, and adding an Authorization Header to it because the initial req object is immutable. But we are not passing this cloned request but the initial request to the `next.handle()` method. This is causing 403 errors, as we are passing an invalid JWT to the backend. As seen in the below image.

- The second one – inside the `handleAuthErrors()` method, if we try to make multiple HTTP calls at the same time, and at that point of time the Auth Token is expired, we will try to refresh the token multiple times.
- In this case, we added an if condition -> we allow the refresh token request only if there isn't an existing refresh token process going on. So in our case, the first request wins and requests a refresh token, but the second request will fail silently.
- To fix this bug, we will add an else condition and will use the `filter()` on the BeahviorSubject until we receive a non null response, we will then accept the first entry in the BehaviorSubject using the `take()` method and finally will use the `switchMap()` to take the new token and use it to make the request.

Token Interceptor before the bug fix changes.

`token-interceptor.ts` **(partial code)**

```
    intercept(req: HttpRequest<any>, next: HttpHandler):
            Observable<HttpEvent<any>> {

            const jwtToken = this.authService.getJwtToken();
            if (jwtToken) {
                this.addToken(req, jwtToken);
            }
            return next.handle(req).pipe(catchError(error => {
                if (error instanceof HttpErrorResponse
                    && error.status === 403) {
                    return this.handleAuthErrors(req, next);
                } else {
                    return throwError(error);
                }
            }));
        }

        addToken(req: HttpRequest<any>, jwtToken: any) {
            return req.clone({
                headers: req.headers.set('Authorization',
                    'Bearer ' + jwtToken)
            });
        }
```

Here is how our `TokenInterceptor` looks like with the above changes:

`token-interceptor.ts`

```
    import { Injectable } from '@angular/core';
    import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent, HttpErrorResponse } from
    '@angular/common/http';
    import { Observable, BehaviorSubject, throwError } from 'rxjs';
    import { AuthService } from './auth/shared/auth.service';
    import { catchError, switchMap, take, filter } from 'rxjs/operators';
    import { LoginResponse } from './auth/login/login-response.payload';
```

```
@Injectable({
    providedIn: 'root'
})
export class TokenInterceptor implements HttpInterceptor {

    isTokenRefreshing = false;
    refreshTokenSubject: BehaviorSubject<any> = new BehaviorSubject(null);

    constructor(public authService: AuthService) { }

    intercept(req: HttpRequest<any>, next: HttpHandler):
        Observable<HttpEvent<any>> {

        if (req.url.indexOf('refresh') !== -1 || req.url.indexOf('login') !== -1) {
            return next.handle(req);
        }
        const jwtToken = this.authService.getJwtToken();

        return next.handle(this.addToken(req, jwtToken)).pipe(catchError(error => {
            if (error instanceof HttpErrorResponse
                && error.status === 403) {
                return this.handleAuthErrors(req, next);
            } else {
                return throwError(error);
            }
        }));
    }

    private handleAuthErrors(req: HttpRequest<any>, next: HttpHandler)
        : Observable<HttpEvent<any>> {
        if (!this.isTokenRefreshing) {
            this.isTokenRefreshing = true;
            this.refreshTokenSubject.next(null);

            return this.authService.refreshToken().pipe(
                switchMap((refreshTokenResponse: LoginResponse) => {
                    this.isTokenRefreshing = false;
                    this.refreshTokenSubject
                        .next(refreshTokenResponse.authenticationToken);
                    return next.handle(this.addToken(req,
                        refreshTokenResponse.authenticationToken));
                })
            )
        } else {
            return this.refreshTokenSubject.pipe(
                filter(result => result !== null),
                take(1),
                switchMap((res) => {
                    return next.handle(this.addToken(req,
                        this.authService.getJwtToken()))
                })
            );
        }
    }

    addToken(req: HttpRequest<any>, jwtToken: any) {
        return req.clone({
            headers: req.headers.set('Authorization',
                'Bearer ' + jwtToken)
        });
    }

}
```

After making this changes, if you try to post a comment, you should see the comment in t

Thank you for visiting. You can now buy me a coffee!

Displaying Username in the Header Section

Now the next task is to display the username in the Header Section, once the user logged in, right now we are always displaying the Login and Signup buttons.

Before making these changes we have to install one library called NgBootstrap, we will use the dropdown menu in this library. Let's install this library using the following command.

Notice that we ran two commands, the first one is npm install `@ng-bootstrap/ng-bootstrap` and the next one is `ng add @angular/localize`, we need to run this command if we want to use NgBootstrap in Angular 9.

After that we have to enable the NgBootstrap in our project by adding the NgbModule to the **app.module.ts** file

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { SignupComponent } from './auth/signup/signup.component';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { LoginComponent } from './auth/login/login.component';
import { NgxWebstorageModule } from 'ngx-webstorage';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { ToastrModule } from 'ngx-toastr';
import { TokenInterceptor } from './token-interceptor';
import { HomeComponent } from './home/home.component';
import { FontAwesomeModule } from '@fortawesome/angular-fontawesome';
import { PostTileComponent } from './shared/post-tile/post-tile.component';
import { VoteButtonComponent } from './shared/vote-button/vote-button.component';
import { SideBarComponent } from './shared/side-bar/side-bar.component';
import { SubredditSideBarComponent } from './shared/subreddit-side-bar/subreddit-
import { CreateSubredditComponent } from './subreddit/create-subreddit/create-sub
import { CreatePostComponent } from './post/create-post/create-post.component';
```

Thank you for visiting. You can now buy me a coffee!

```
import { ListSubredditsComponent } from './subreddit/list-subreddits/list-subreddits.component';
import { EditorModule } from '@tinymce/tinymce-angular';
import { ViewPostComponent } from './post/view-post/view-post.component';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';


@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    SignupComponent,
    LoginComponent,
    HomeComponent,
    PostTileComponent,
    VoteButtonComponent,
    SideBarComponent,
    SubredditSideBarComponent,
    CreateSubredditComponent,
    CreatePostComponent,
    ListSubredditsComponent,
    ViewPostComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    HttpClientModule,
    NgxWebstorageModule.forRoot(),
    BrowserAnimationsModule,
    ToastrModule.forRoot(),
    FontAwesomeModule,
    EditorModule,
    NgbModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: TokenInterceptor,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Now we are ready to make our changes, copy the below HTML, CSS and Typescript files to the project.

`header-component.html`

```
<header>
    <nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
        <div class="flex-grow-1">
            <a aria-label="Home" class="logo" routerLink="/">
                <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20" class="reddit-icon-svg">
                    <g>
                        <circle fill="#FF4500" cx="10" cy="10" r="10"></circle>
                        <path fill="#FFF"
                            d="M16.67,10A1.46,1.46,0,0,0,14.2,9a7.12,7.12,0,0,0-3.85-
1.23L11,4.65,13.14,5.1a1,1,0,1,0,.13-0.61L10.82,4a0.31,0.31,0,0,0-.37.24L9.71,7.71a7.14,7.14,0,0,0-
3.9,1.23A1.46,1.46,0,1,0,4.2,11.33a2.87,2.87,0,0,0,0,.44c0,2.24,2.61,4.06,5.83,4.06s5.83-1.82,5.83-
4.06a2.87,2.87,0,0,0,0-.44A1.46,1.46,0,0,0,16.67,10Zm-
10,1a1,1,0,1,1,1,1A1,1,0,0,1,6.67,11Zm5.81,2.75a3.84,3.84,0,0,1-2.47.77,3.84,3.84,0,0,1
2.47-.77,0.27,0.27,0,0,1,.38-
0.38A3.27,3.27,0,0,0,10,14a3.28,3.28,0,0,0,2.09-.61A0.27,0.27,0,1,1,12.48,13.79Zm
1A1,1,0,0,1,12.29,12.08Z">
```

```
                        </path>
                    </g>
                </svg>
                <span class="reddit-text">
                    Spring Reddit Clone
                </span>
            </a>
        </div>
        <div class="flex-grow-1 float-right">
            <div *ngIf="isLoggedIn" ngbDropdown class="float-right">
                <div ngbDropdownMenu aria-labelledby="dropdownBasic1">
                    <button (click)="goToUserProfile()" ngbDropdownItem>Profile</button>
                    <button (click)="logout()" ngbDropdownItem>Logout</button>
                </div>
                <button class="userdetails" id="dropdownBasic1" ngbDropdownToggle>
                    <img class="account-icon"
src="https://www.redditstatic.com/avatars/avatar_default_08_D4E815.png">
                    {{username}}
                </button>
            </div>
            <div *ngIf="!isLoggedIn">
                <a routerLink="/sign-up" class="float-right sign-up mr-2">Sign up</a>
                <a routerLink="/login" class="float-right login mr-2">Login</a>
            </div>
        </div>
    </nav>
</header>
```

header.component.css

```
header{
    border-radius: 1px solid;
}
.reddit-icon-svg{
    height: 50px;
    padding: 8px 8px 8px 0;
    width: 50px;
}

.reddit-text{
    font-weight: 700;
    height: 50px;
    width: 50px;
}
.logo{
    text-decoration: none;
}

.login, .sign-up{
    background-color: transparent;
    border-color: #0079D3;
    color: #0079D3;
    fill: #0079D3;
    border: 1px solid;
    border-radius: 4px;
    box-sizing: border-box;
    text-align: center;
    letter-spacing: 1px;
    text-decoration: none;
    font-size: 12px;
    font-weight: 700;
    letter-spacing: .5px;
    line-height: 24px;
    text-transform: uppercase;
    padding: 3px 16px;
```

```
        opacity: 1;
    }

    .sign-up{
        background-color: #0079D3;
        border-color: #0079D3;
        color: aliceblue;
    }

    .sign-up:hover{
        opacity: 0.6;
    }

    .userdetails{
        background-color: transparent;
        border-color: #0079D3;
        color: #0079D3;
        fill: #0079D3;
        border: 1px solid;
        border-radius: 4px;
        box-sizing: border-box;
        text-align: center;
        letter-spacing: 1px;
        text-decoration: none;
        font-size: 12px;
        font-weight: 700;
        letter-spacing: .5px;
        line-height: 24px;
        text-transform: uppercase;
        padding: 3px 16px;
        opacity: 1;
        border: 0px;
    }

    .userdetails:hover{
        border: 1px solid;
    }

    .dropdown-item{
        background-color: #f8f9fa;
        font-size: 14px;
        font-weight: 500;
        line-height: 18px;
        display: inline-block;
        vertical-align: middle;
    }

    .dropdown-item:hover{
        background-color: #0079D3;
    }

    .dropdown-menu{
        background-color: #f8f9fa;
    }

    .account-icon{
        border-radius: 4px;
        float: left;
        margin-right: 5px;
        max-height: 24px;
        max-width: 24px;
    }
```

header-component.ts

```
import { Component, OnInit } from '@angular/core';
import { faUser } from '@fortawesome/free-solid-svg-icons';
import { AuthService } from '../auth/shared/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {
  faUser = faUser;
  isLoggedIn: boolean;
  username: string;

  constructor(private authService: AuthService, private router: Router) { }

  ngOnInit() {
    this.isLoggedIn = this.authService.isLoggedIn();
    this.username = this.authService.getUserName();
  }

  goToUserProfile() {
    this.router.navigateByUrl('/user-profile/' + this.username);
  }
}
```

If you have a look at the header section now inside the application, this is how it should look like.

It looks good, right? Now let's add some functionality to display the User Profile.

Display User Profile

Let's create a new Component called UserProfileComponent inside the auth folder.

And copy the below code

user-profile.component.html

```
<div class="container">
  <div>
    Welcome <b>{{name}}</b>.<br /> You have posted <b>{{postLength}}</b> time(s) and commented
    <b>{{commentLength}}</b> time(s).
    You can check your post and comment history below.
  </div>
  <hr />
  <div>
    Your Posts:
  </div>
  <app-post-tile [posts]="posts"></app-post-tile>
  <hr />
  <div>
    Your Comments:
  </div>
  <div *ngFor="let comment of comments">
    <div class="comment">
      <div class="username">
        <a routerLink="/user/comment.username">{{comment.userName}}</a>
      </div>
      <div>
        <p>{{comment.duration}}</p>
      </div>
      <b>{{comment.text}}</b>
    </div>
    <hr />
  </div>
</div>
```

user-profile.component.ts

```
import { Component, OnInit } from '@angular/core';
import { PostService } from 'src/app/shared/post.service';
import { ActivatedRoute } from '@angular/router';
import { CommentService } from 'src/app/comment/comment.service';
import { PostModel } from 'src/app/shared/post-model';
import { CommentPayload } from 'src/app/comment/comment.payload';

@Component({
  selector: 'app-user-profile',
  templateUrl: './user-profile.component.html',
  styleUrls: ['./user-profile.component.css']
})
export class UserProfileComponent implements OnInit {
  name: string;
```

Thank you for visiting. You can now buy me a coffee!

```
  posts: PostModel[];
  comments: CommentPayload[];
  postLength: number;
  commentLength: number;

  constructor(private activatedRoute: ActivatedRoute, private postService: PostService,
    private commentService: CommentService) {
    this.name = this.activatedRoute.snapshot.params.name;

    this.postService.getAllPostsByUser(this.name).subscribe(data => {
      this.posts = data;
      this.postLength = data.length;
    });
    this.commentService.getAllCommentsByUser(this.name).subscribe(data => {
      this.comments = data;
      this.commentLength = data.length;
    });
  }

  ngOnInit(): void {
  }

}
```

Before testing this code, we have to make small refactoring to our PostTileComponent.

In our UserProfileComponent, we want to display the posts which are created by the User, for this, we can re-use the PostTileComponent we created in the previous article.

But in `PostTileComponent`, if you observe we are reading all the posts and displaying them, instead of reading all the posts inside the constructor we should pass on the posts which we want to see inside this component as an input, let's refactor the `post-tile.component.ts file`

```
import { Component, OnInit, ViewEncapsulation, Input } from '@angular/core';
import { PostService } from '../post.service';
import { PostModel } from '../post-model';
import { faComments } from '@fortawesome/free-solid-svg-icons';
import { Router } from '@angular/router';

@Component({
  selector: 'app-post-tile',
  templateUrl: './post-tile.component.html',
  styleUrls: ['./post-tile.component.css'],
  encapsulation: ViewEncapsulation.None,
})
export class PostTileComponent implements OnInit {

  faComments = faComments;
  @Input()
  posts: PostModel[];

  constructor(private router: Router) { }

  ngOnInit(): void {
  }

  goToPost(id: number): void {
    this.router.navigateByUrl('/view-post/' + id);
  }
}
```

So we added an @Input to our component with variable posts which is of type PostModel[], now we have to move the logic to read all the posts to the home page component,let's update the home.component.ts file with the below code

```
import { Component, OnInit } from '@angular/core';
import { PostModel } from '../shared/post-model';
import { PostService } from '../shared/post.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  posts: Array<PostModel> = [];

  constructor(private postService: PostService) {
    this.postService.getAllPosts().subscribe(post => {
      this.posts = post;
    });
  }

  ngOnInit(): void {
  }

}
```

Before testing our implementation there is another small bug we have to fix in our backend, inside the PostMapper interface, make sure to add the mapping for the user field, if not the created Posts won't be assigned any user.

PostMapper.java

```
.......
    @Mapping(target = "createdDate", expression = "java(java.time.Instant.now())")
    @Mapping(target = "description", source = "postRequest.description")
    @Mapping(target = "subreddit", source = "subreddit")
    @Mapping(target = "voteCount", constant = "0")
    @Mapping(target = "user", source = "user")
    public abstract Post map(PostRequest postRequest, Subreddit subreddit, User user);
```

We are able to see the Posts and Comments submitted by the user.

Conclusion

So that is it, we came to end of the article Full Stack Reddit Clone with Spring boot and Angular – Part 16, in the next article we will see how to submit votes and implement Logout in our Angular Application.

I will see you in the next article, until then Happy Coding and Stay Safe 🙂

Icons used in this article are made by Flat Icons, Smashicons, Eucalyp from www.flaticon.com

About the author

Sai Upadhyayula