PROJECT REPORT ON BLOGGING APPLICATION

First of all, I would like to thank my lecturer Mr. Cittaranjan Ghosh for helping me to acquire knowledge of "Java Programming Language". At the same time, he gave me the opportunity to learn something new related to our module like spring MVC, spring Boot ,MySQL Database, Rest API, Hibernate etc.

This assignment is based on developing an Blogging application using "Spring Boot" so that it will become more users friendly to interact.

Besides, I also added all the modules and records in this documentation. Let's see the Modules

Name: Lakshmi Narayanan's

Batchcode:7671

Batch Center:Chennai-chromepet

Abstract

Web application development has played an important role in software Engineering. Model- View-Controller (MVC) pattern lays a foundation for developing web applications. The MVC architecture separates an application into different business logic (data presentation, data management and request handling). The Spring Framework is an application framework used byJava application and there are extensions for building web applications

[1]. Spring Boot is a framework tool designed to simplify the initialization of Spring, which makes it easy to create stand-alone, production-grade Spring based applications
[2]. Hibernate is an Object-Relational Mapping tool which maps the database tables to the Object classes. The goal of this project is to develop a personal blog that can be used to write and post articles, pictures and codes by the administrator. The blog also allows general users to read and comment on the blogs. The architecture of the proposed system will be based on Spring Boot and Hibernate

Table of Contents

Detailed explanation of steps involved in my project

- *Setup Spring boot Project*
- *Project Structure*
- *Set up MySQL Database*
- *Configure Database, Hibernate & JPA in our project*
- *Create Domain Entities in our project*
- *Create Repositories for our Post and User classes*
- *Revisiting the Project Structure*
- *Running the Application*

Chapter 2: Technology Overview

This chapter presents information about the Spring, Spring Boot and Hibernate frameworks – their brief introduction, features and architectures. The first section is about Spring and Spring Boot and the second section explains the Hibernate framework.

2.1 Spring and Spring Boot

Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications.

2.1.1 Spring Architecture

The architecture of Spring is shown in Figure 1. This figure is adapted from 'Introduction to Spring Framework' [3].

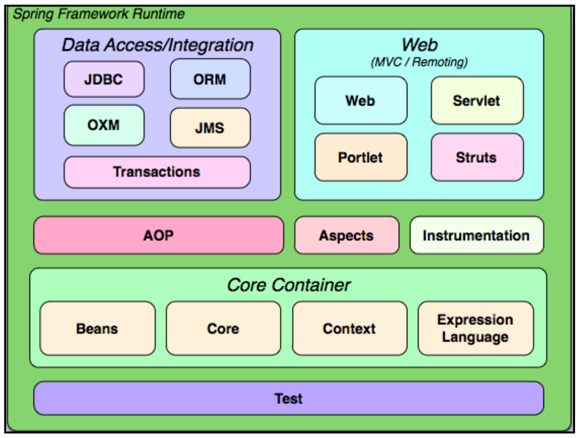


Figure 1: Overview of the Spring Framework

Spring framework consists of the following components [3]:

- **Core Container:** The Core container consists of the Core, Beans, Context and Expression Language modules**,** The Core and Beans modules provide the fundamental parts of the framework. The Context module builds on the solid base provided by the Core and Beans modules. The Expression Language module provides a powerful expression language for querying and manipulating an object graph at runtime.

- **Data Access/Integration:** The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules. The JDBC module provides a JDBC-abstraction layer that removes the need to do tedious JDBC coding and parsing of database-vendor specific error codes. The ORM module provides integration layers for popular object-relational mapping APIs.
- **Web:** Spring's Web module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the Inverse of Control (IoC) container using servlet listeners and a web-oriented application context. It also contains the web-related parts of Spring's remoting support.
- **AOP and Instrumentation:** Spring's AOP module provides an AOP Alliance-compliant aspect-oriented programming implementation allowing you to define, for example, method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.

. The Administrator subsystem provides the following functionalities:

Admin Login: An administrator can login with username and password to manage the content.

Manage Blog: An administrator can publish a new blog, modify the existing blogs, delete a blog and search for a blog.

Manage Types: An administrator can create a new archive, modify the existing archives, delete archives and search for an archive.

Manage Tags: An administrator can create a new tag, modify the existing tagged blogs, delete tags and search for a tag.

Brief Description The Manage Blog use case enables the Blog System Administrator to manage the blog content.
Step-by-Step Description 1. Allow the following modifications to the current blog system: • Publish a new blog • Modify a current blog • Delete a current blog • Search for a blog

Figure 5. Login use Case for the Blog System

Table 1: *Login* use Case Description for the Blog System

Brief Description The Login use case enables the Blog System Administrator to login into the system.
Step-by-Step Description 1. Enter the Username and Password at the login screen. 2. Validate the Username and Password entered by the user to display the administrator screen. Wrong Username or Password will stay on the login screen with error message.

Figure 7: *Manage Type* use Case for the Blog System

Table 2: Manage Archive use Case Description for the Blog System

Brief Description The Manage Types use case enables the Blog System Administrator to manage the types for the blog.
Step-by-Step Description 1. Allow the following modifications to the current blog system: • Create a new type • Change a current blog article to a different type • Delete a current type

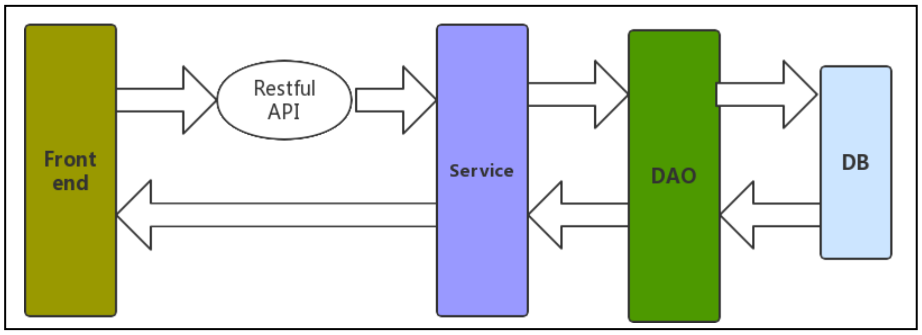


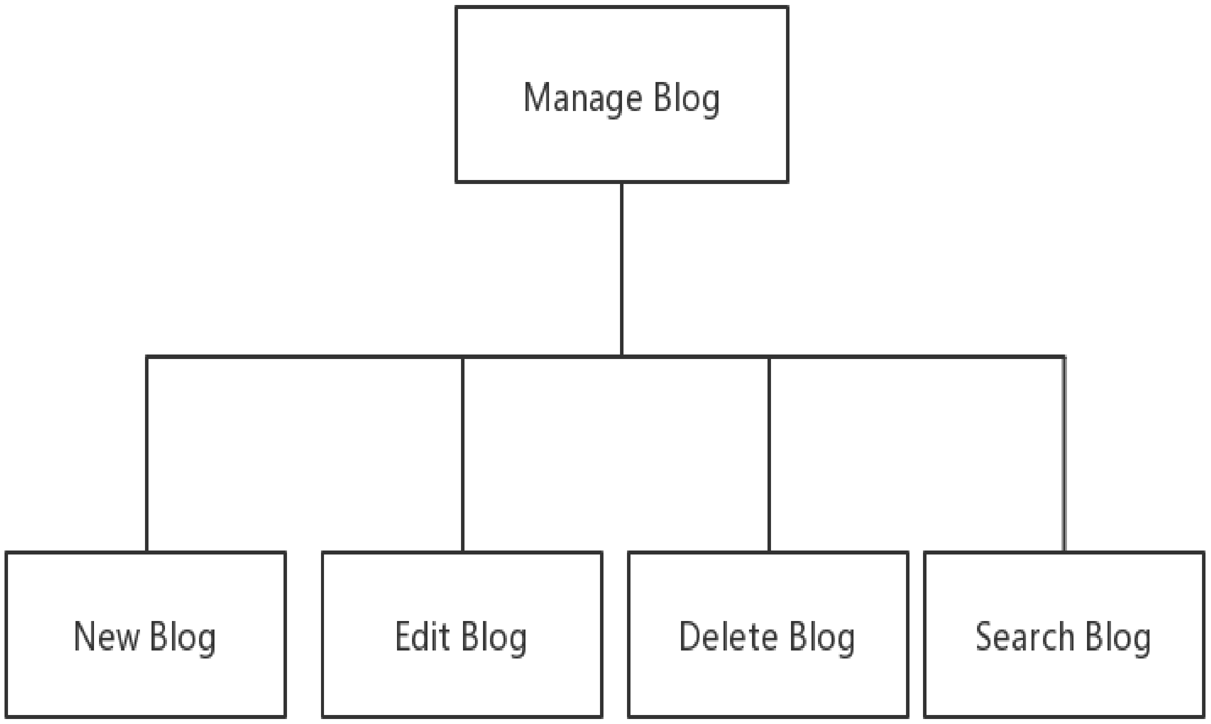
Figure 10: System Architecture

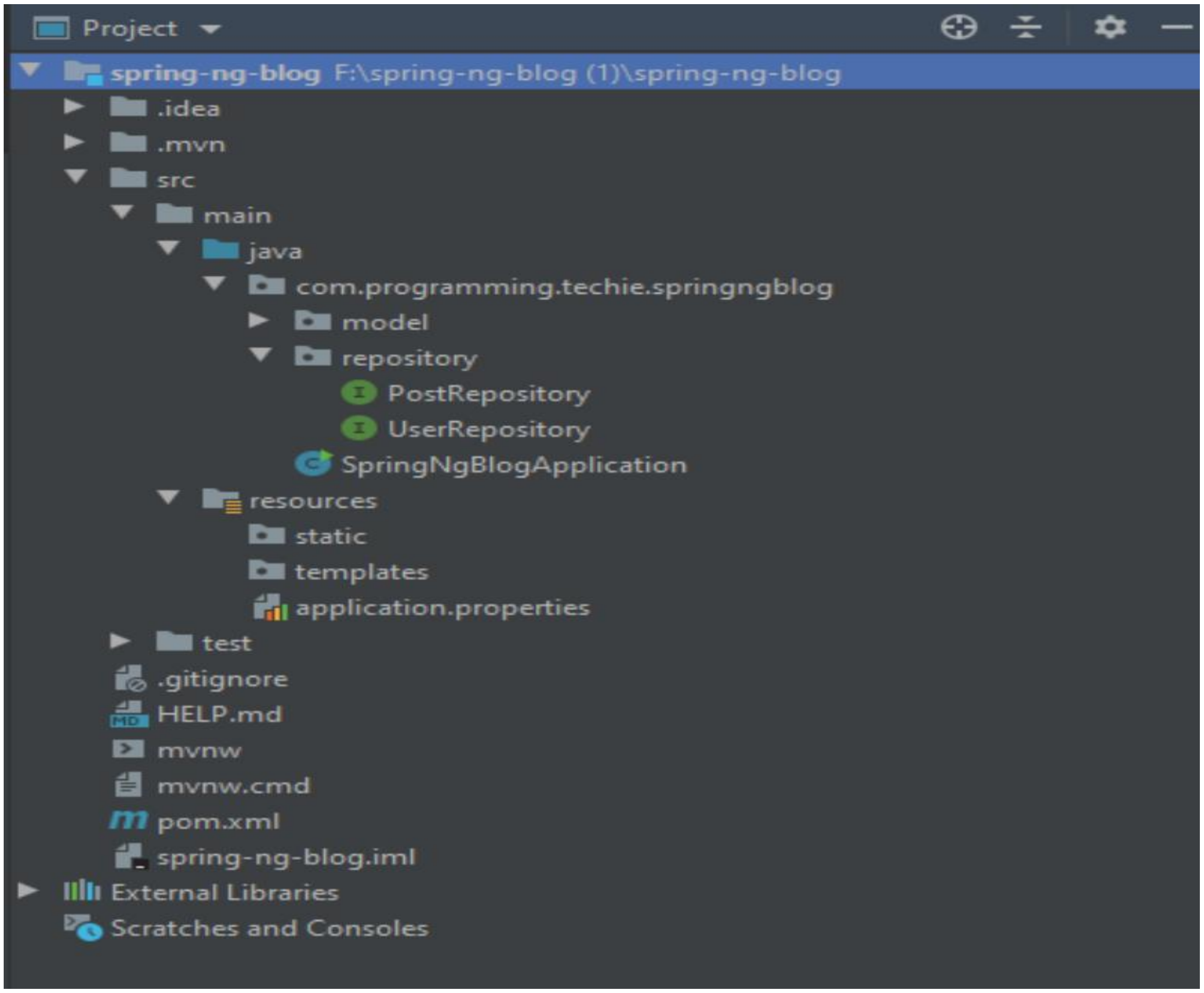
Design of the frontend. This layer will be the user interface. It is responsible for displaying information to user.

RESTful API. REST is the underlying architectural principle of the web. The amazing thing about the web is the fact that clients (browsers) and servers can interact in complex ways without the client knowing anything beforehand about the server and the resources it hosts. The key constraint is that the server and client must both agree on the *media* used, which in the case of the web is HTML. An API that adheres to the principles of *REST* does not require the client to know anything about the structure of the API. Rather, the server needs to provide whatever information the client needs to interact with the service [4].

Design of the service. This would be the service layer for the application. The application logic will be presented in this layer and it will be responsible for answering the call from the RESTful API and interacting with the DAO layer to receive data. This layer would be implemented by the Spring Boot framework.

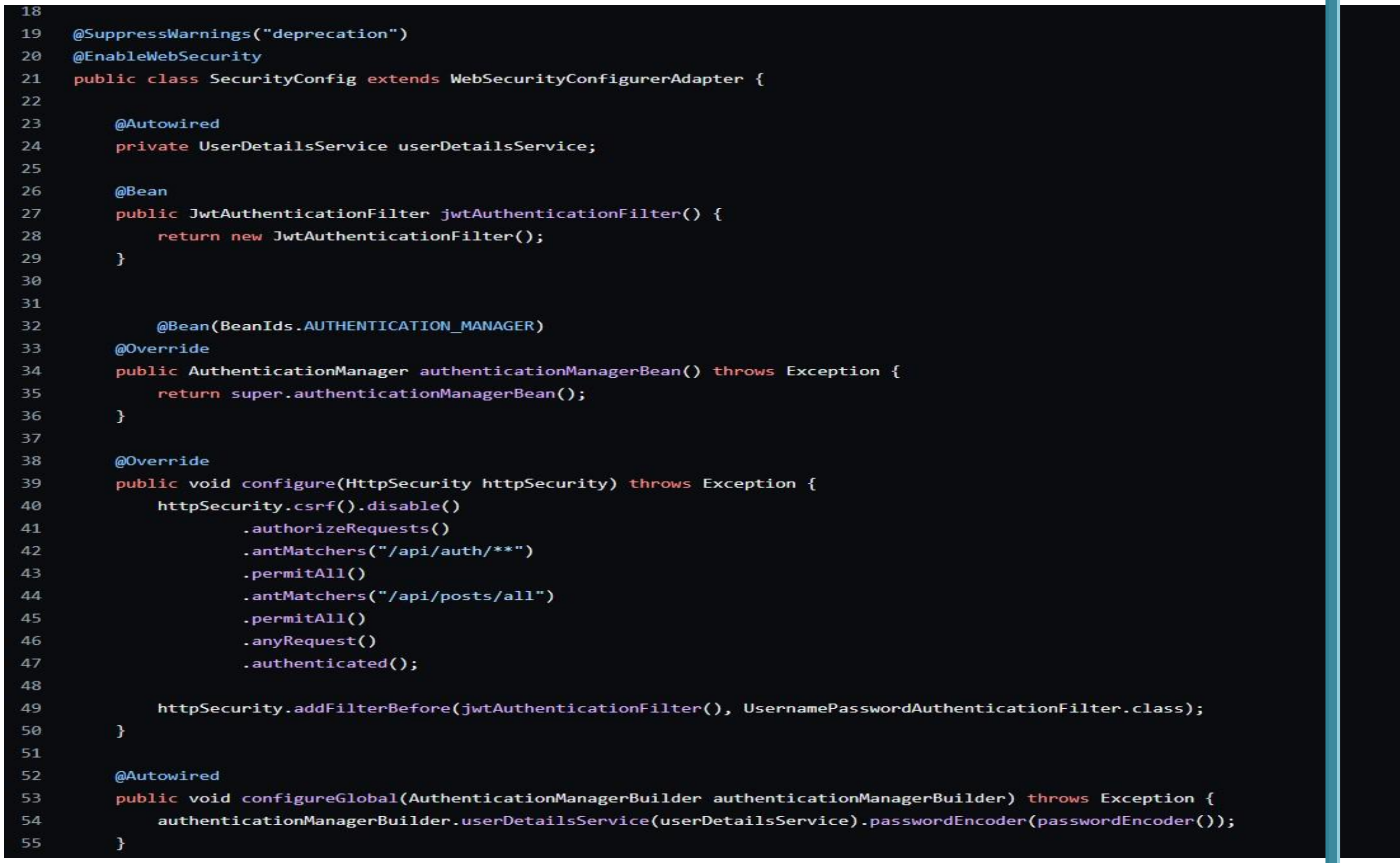
Design of the DAO. This layer is responsible for adding and inserting entity objects into database, updating entity objects in database, and selecting entity objects from database.


Manage Blog
New Blog
Edit Blog
Delete Blog
Search Blog



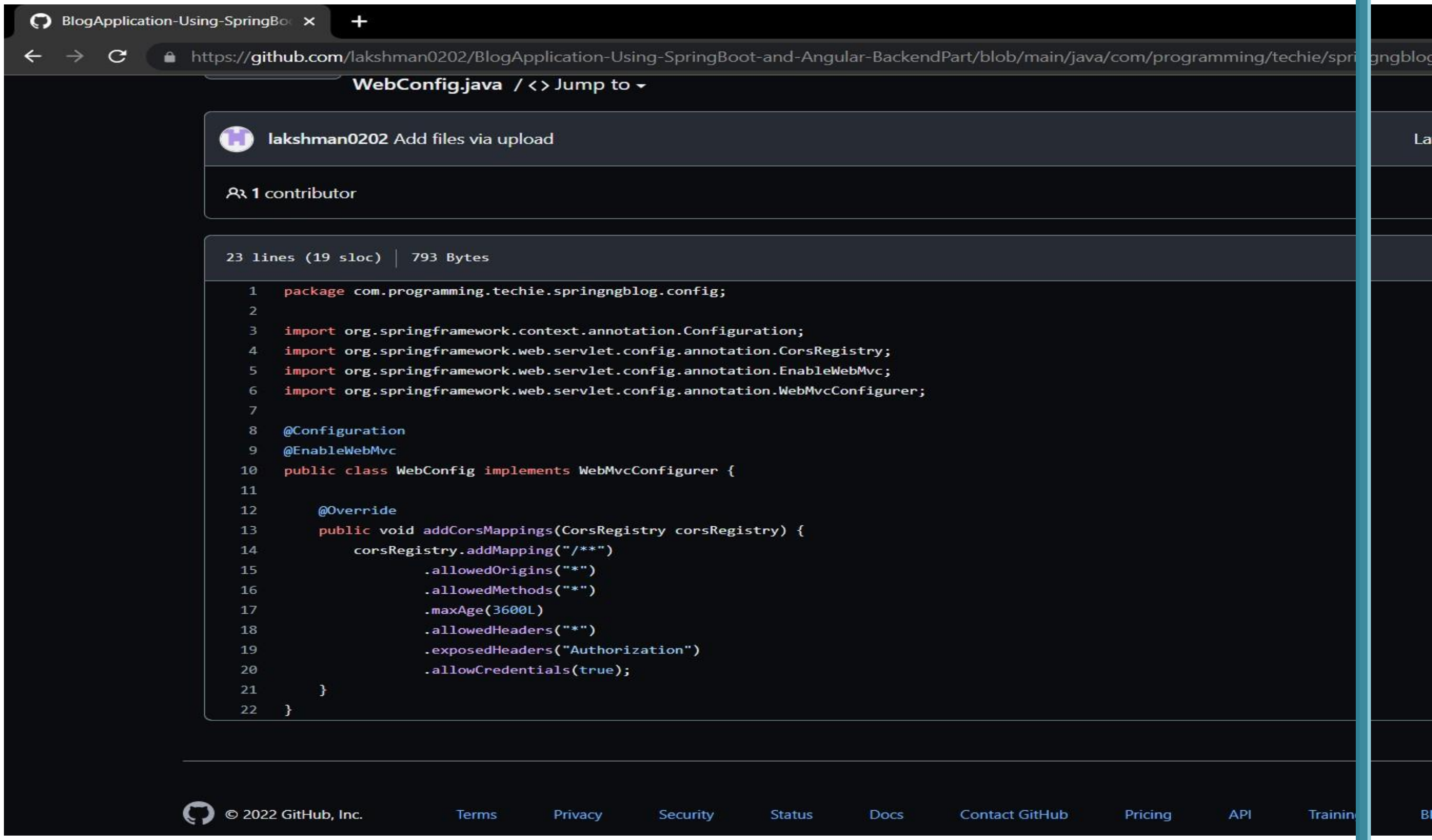
```
spring-ng-blog  F:\spring-ng-blog (1)\spring-ng-blog
  .idea
  .mvn
  src
    main
      java
        com.programming.techie.springngblog
          model
          repository
            PostRepository
            UserRepository
          SpringNgBlogApplication
      resources
        static
        templates
        application.properties
    test
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  spring-ng-blog.iml
External Libraries
Scratches and Consoles
```

Security configuration.



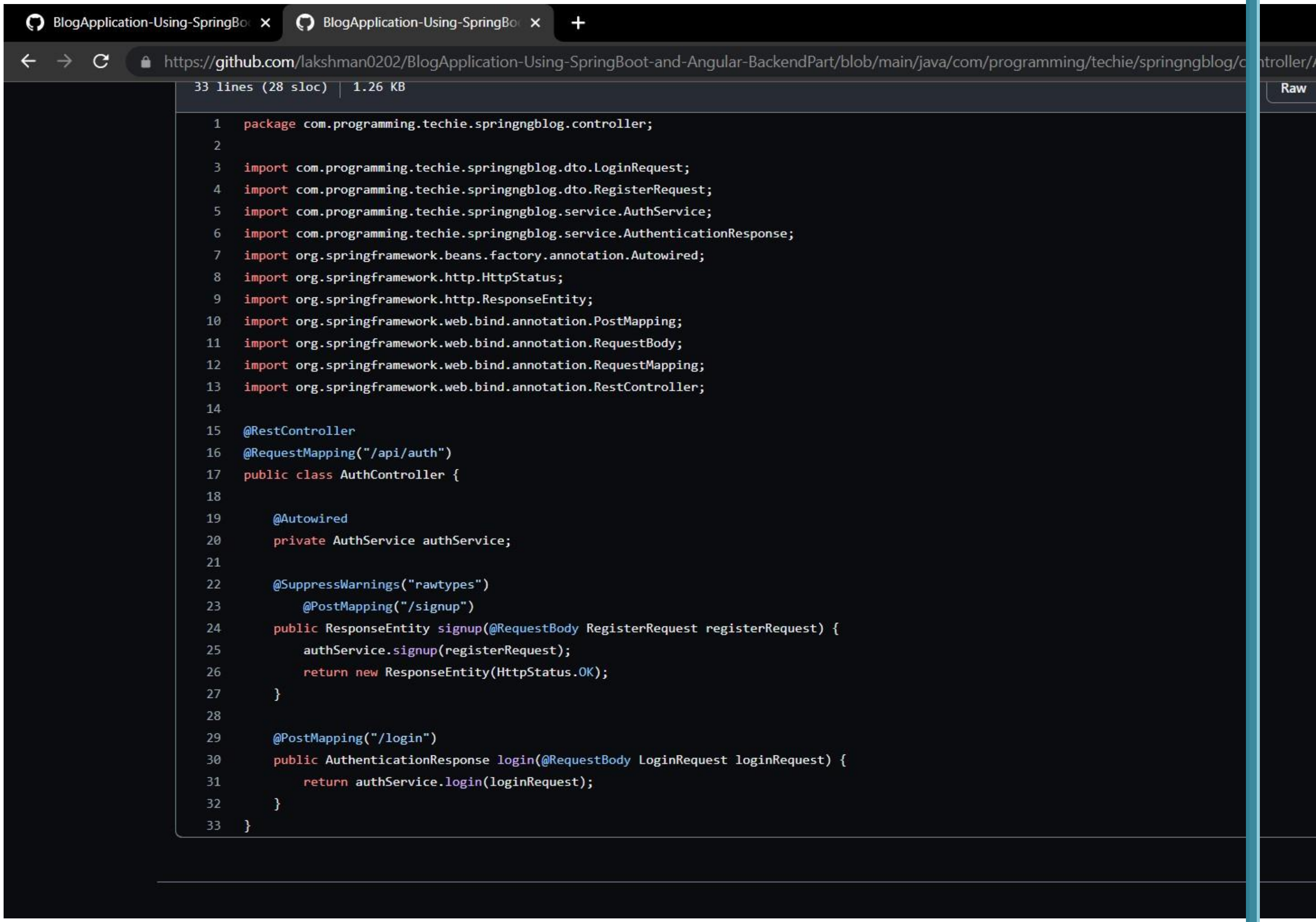
```java
@SuppressWarnings("deprecation")
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    public JwtAuthenticationFilter jwtAuthenticationFilter() {
        return new JwtAuthenticationFilter();
    }


        @Bean(BeanIds.AUTHENTICATION_MANAGER)
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    public void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable()
                .authorizeRequests()
                .antMatchers("/api/auth/**")
                .permitAll()
                .antMatchers("/api/posts/all")
                .permitAll()
                .anyRequest()
                .authenticated();

        httpSecurity.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder authenticationManagerBuilder) throws Exception {
        authenticationManagerBuilder.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }
```

Web configuration


BlogApplication-Using-SpringBo
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/spri gngblog
WebConfig.java / <> Jump to
lakshman0202 Add files via upload
1 contributor
23 lines (19 sloc) | 793 Bytes
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API

```java
package com.programming.techie.springngblog.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry corsRegistry) {
        corsRegistry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("*")
                .maxAge(3600L)
                .allowedHeaders("*")
                .exposedHeaders("Authorization")
                .allowCredentials(true);
    }
}
```

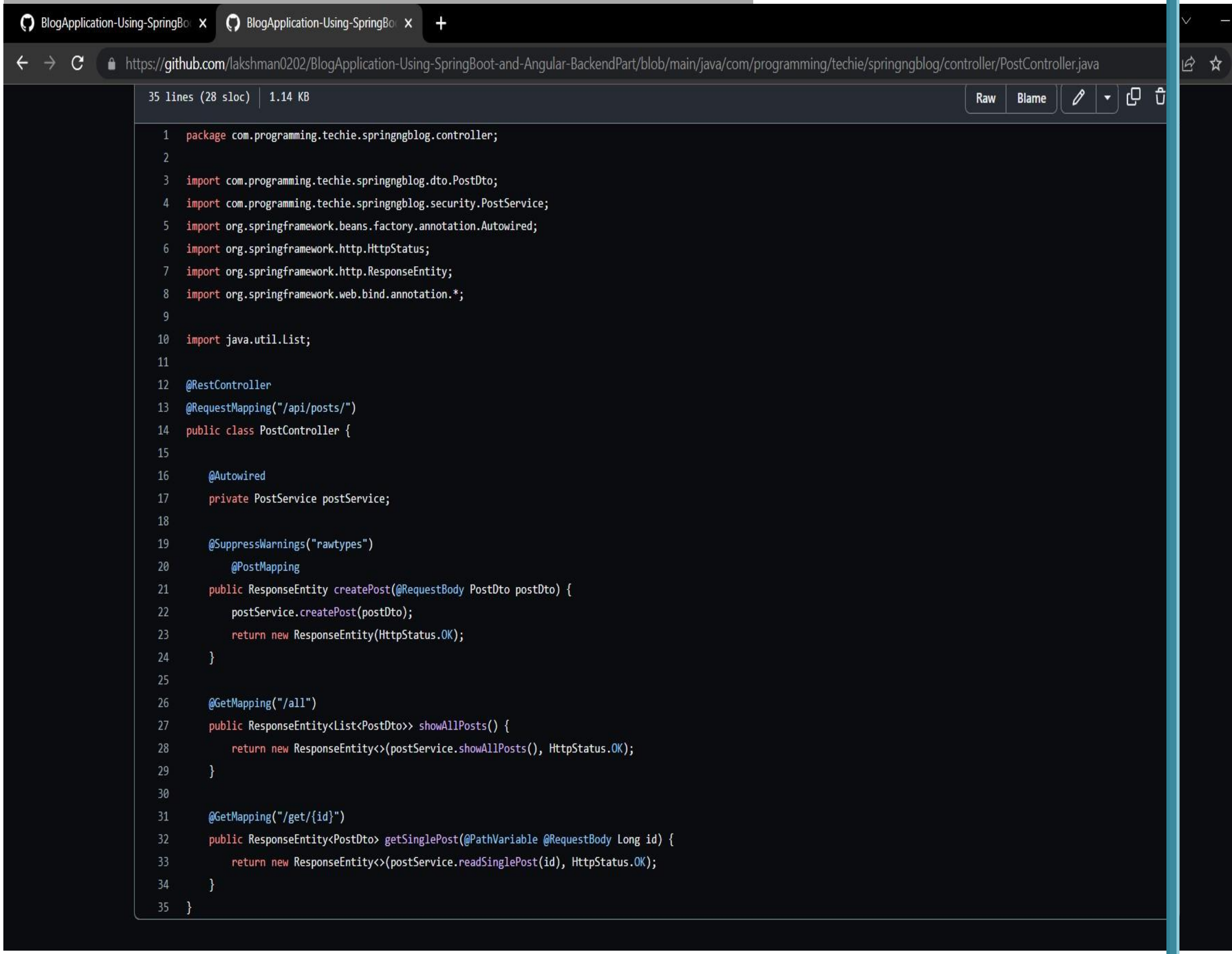
Authuntication controller


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/c troller/
Raw

33 lines (28 sloc) | 1.26 KB

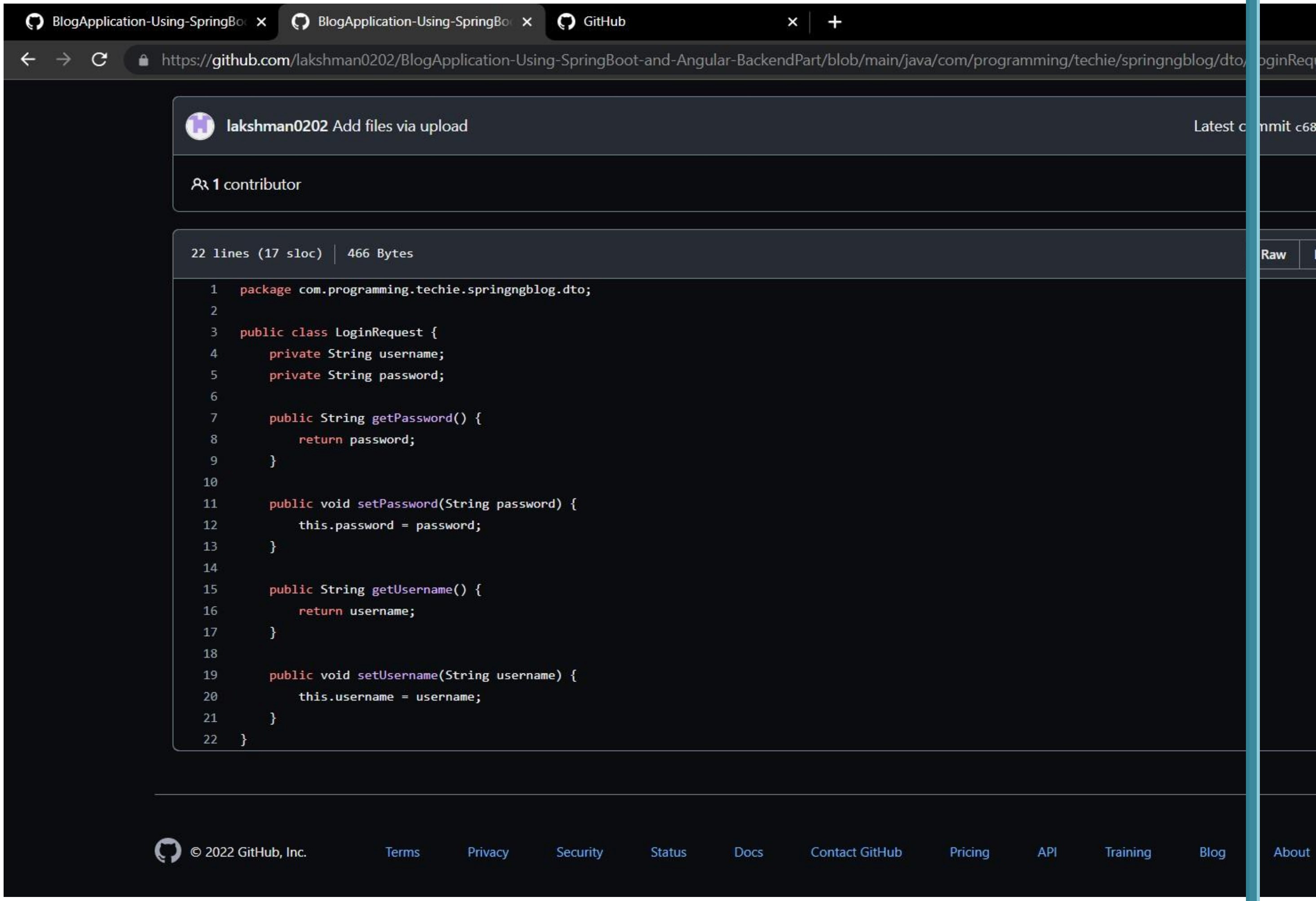
```java
package com.programming.techie.springngblog.controller;

import com.programming.techie.springngblog.dto.LoginRequest;
import com.programming.techie.springngblog.dto.RegisterRequest;
import com.programming.techie.springngblog.service.AuthService;
import com.programming.techie.springngblog.service.AuthenticationResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    @Autowired
    private AuthService authService;

    @SuppressWarnings("rawtypes")
        @PostMapping("/signup")
    public ResponseEntity signup(@RequestBody RegisterRequest registerRequest) {
        authService.signup(registerRequest);
        return new ResponseEntity(HttpStatus.OK);
    }

    @PostMapping("/login")
    public AuthenticationResponse login(@RequestBody LoginRequest loginRequest) {
        return authService.login(loginRequest);
    }
}
```

Post controller


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/controller/PostController.java
35 lines (28 sloc) | 1.14 KB
Raw
Blame

```java
package com.programming.techie.springngblog.controller;

import com.programming.techie.springngblog.dto.PostDto;
import com.programming.techie.springngblog.security.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

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

    @Autowired
    private PostService postService;

    @SuppressWarnings("rawtypes")
        @PostMapping
    public ResponseEntity createPost(@RequestBody PostDto postDto) {
        postService.createPost(postDto);
        return new ResponseEntity(HttpStatus.OK);
    }

    @GetMapping("/all")
    public ResponseEntity<List<PostDto>> showAllPosts() {
        return new ResponseEntity<>(postService.showAllPosts(), HttpStatus.OK);
    }

    @GetMapping("/get/{id}")
    public ResponseEntity<PostDto> getSinglePost(@PathVariable @RequestBody Long id) {
        return new ResponseEntity<>(postService.readSinglePost(id), HttpStatus.OK);
    }
}
```

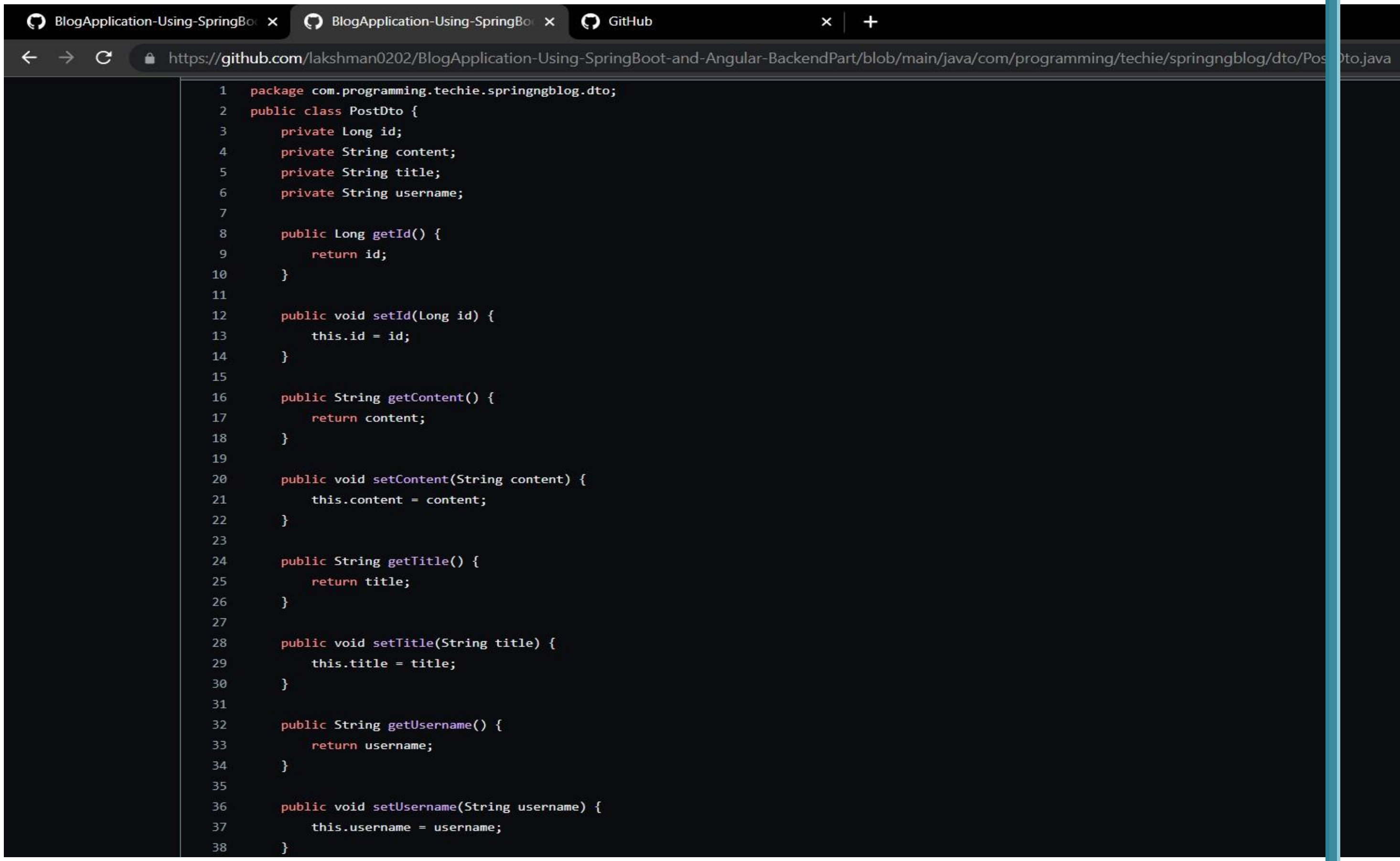
login request__ ____


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/dto/ oginReq
lakshman0202 Add files via upload
Latest c mmit c68
1 contributor
22 lines (17 sloc) | 466 Bytes
Raw

```java
package com.programming.techie.springngblog.dto;

public class LoginRequest {
    private String username;
    private String password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

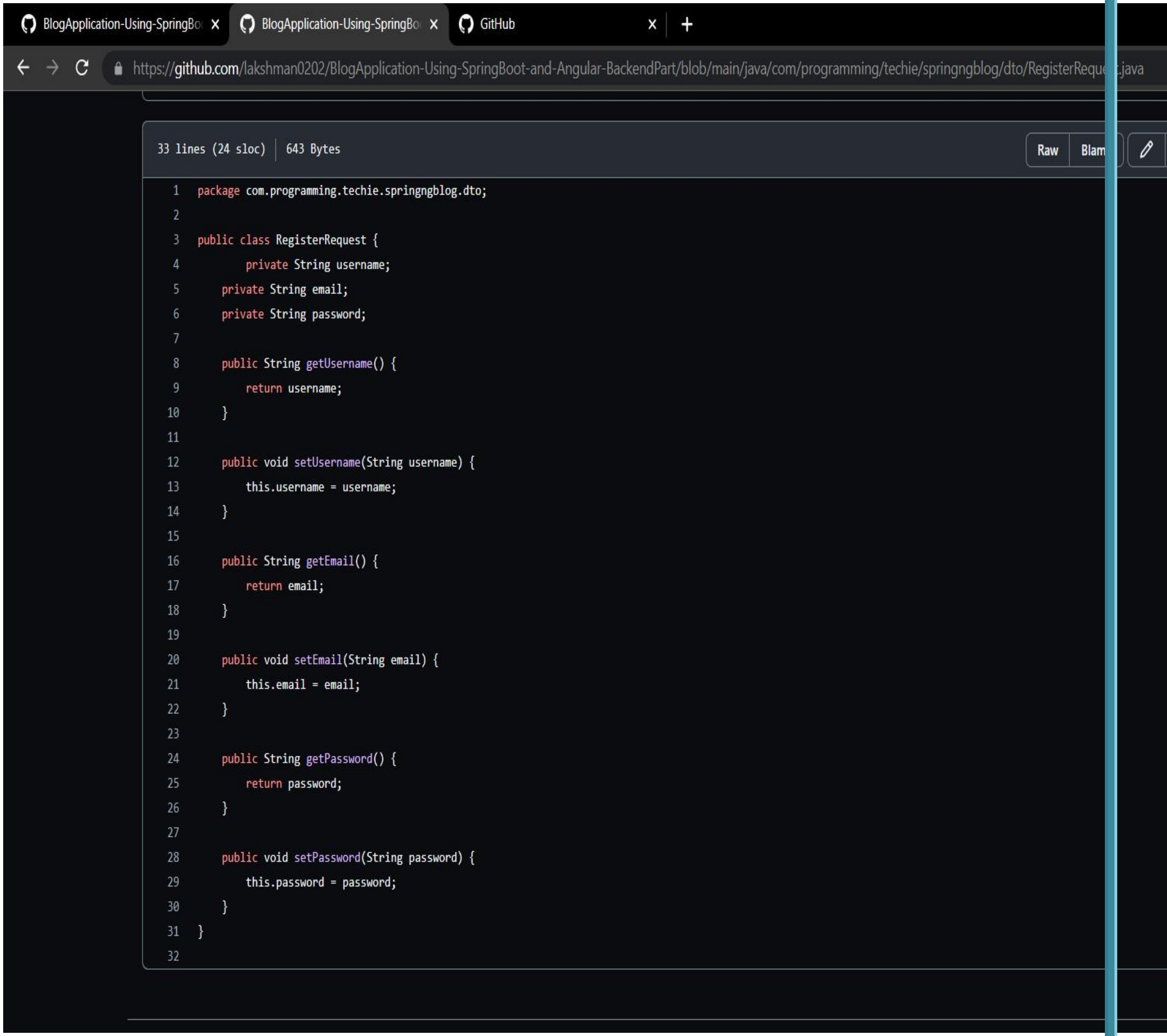
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

Post DTO


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/dto/Pos to.java

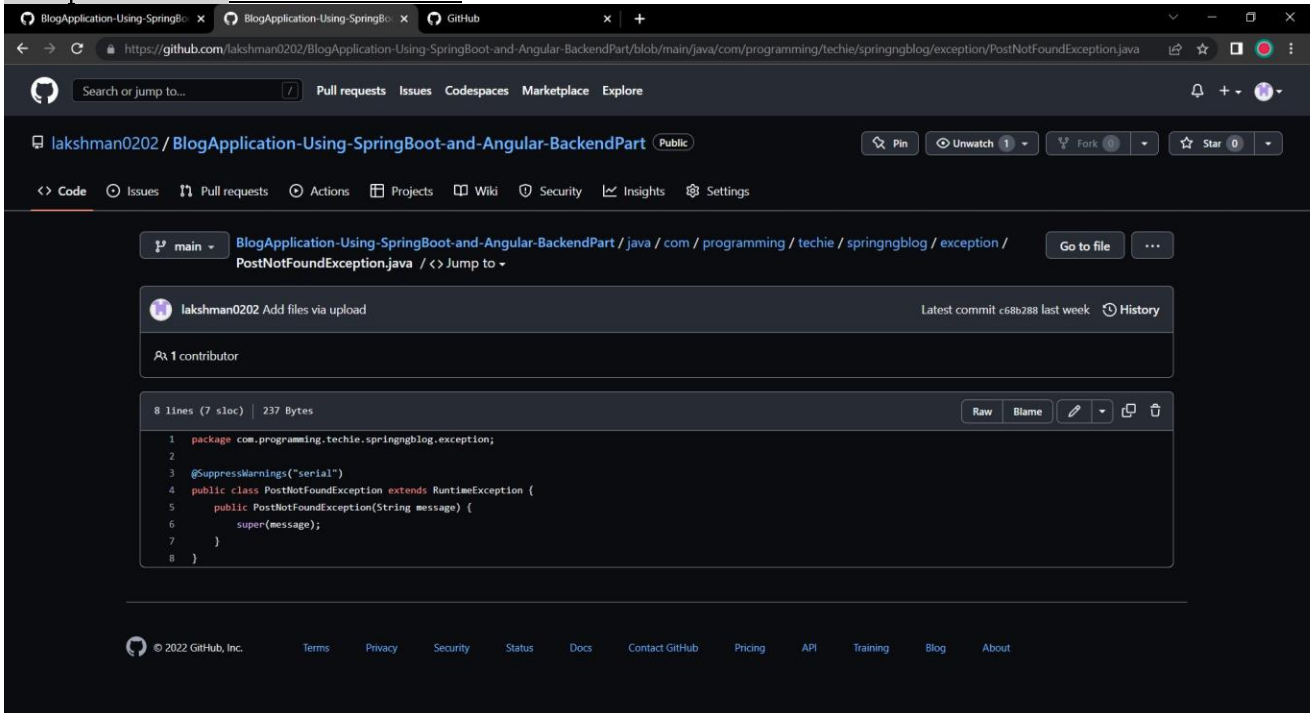
```java
package com.programming.techie.springngblog.dto;
public class PostDto {
    private Long id;
    private String content;
    private String title;
    private String username;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
```

Register request


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/dto/RegisterReque.java
33 lines (24 sloc) | 643 Bytes
Raw
Blam

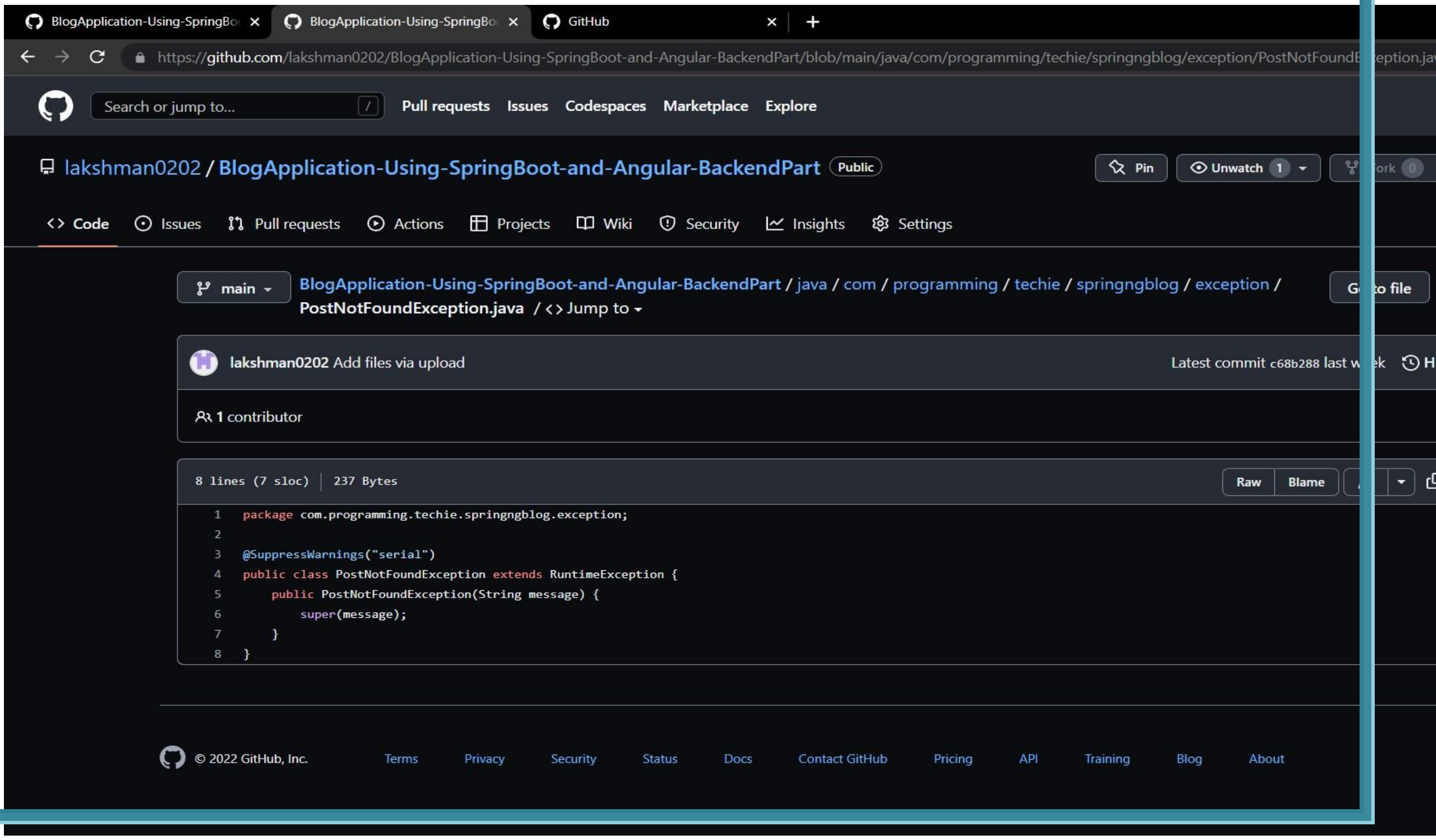
```java
package com.programming.techie.springngblog.dto;

public class RegisterRequest {
        private String username;
    private String email;
    private String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}

```

Exception classes


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/exception/PostNotFoundException.java
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
lakshman0202 / BlogApplication-Using-SpringBoot-and-Angular-BackendPart
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
BlogApplication-Using-SpringBoot-and-Angular-BackendPart / java / com / programming / techie / springngblog / exception / PostNotFoundException.java / <> Jump to
Go to file
Latest commit c68b288 last week
History
1 contributor
8 lines (7 sloc) | 237 Bytes
Raw
Blame
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

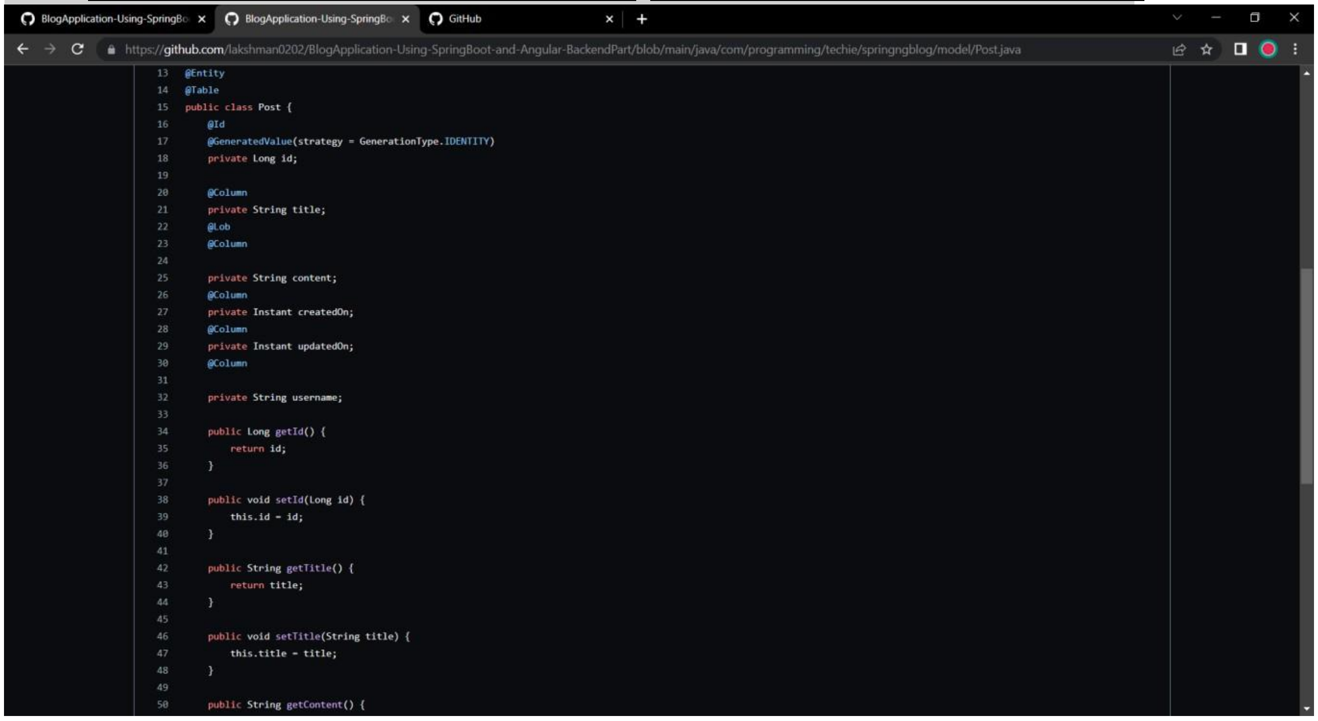
lakshman0202 Add files via upload

```java
package com.programming.techie.springngblog.exception;

@SuppressWarnings("serial")
public class PostNotFoundException extends RuntimeException {
    public PostNotFoundException(String message) {
        super(message);
    }
}
```

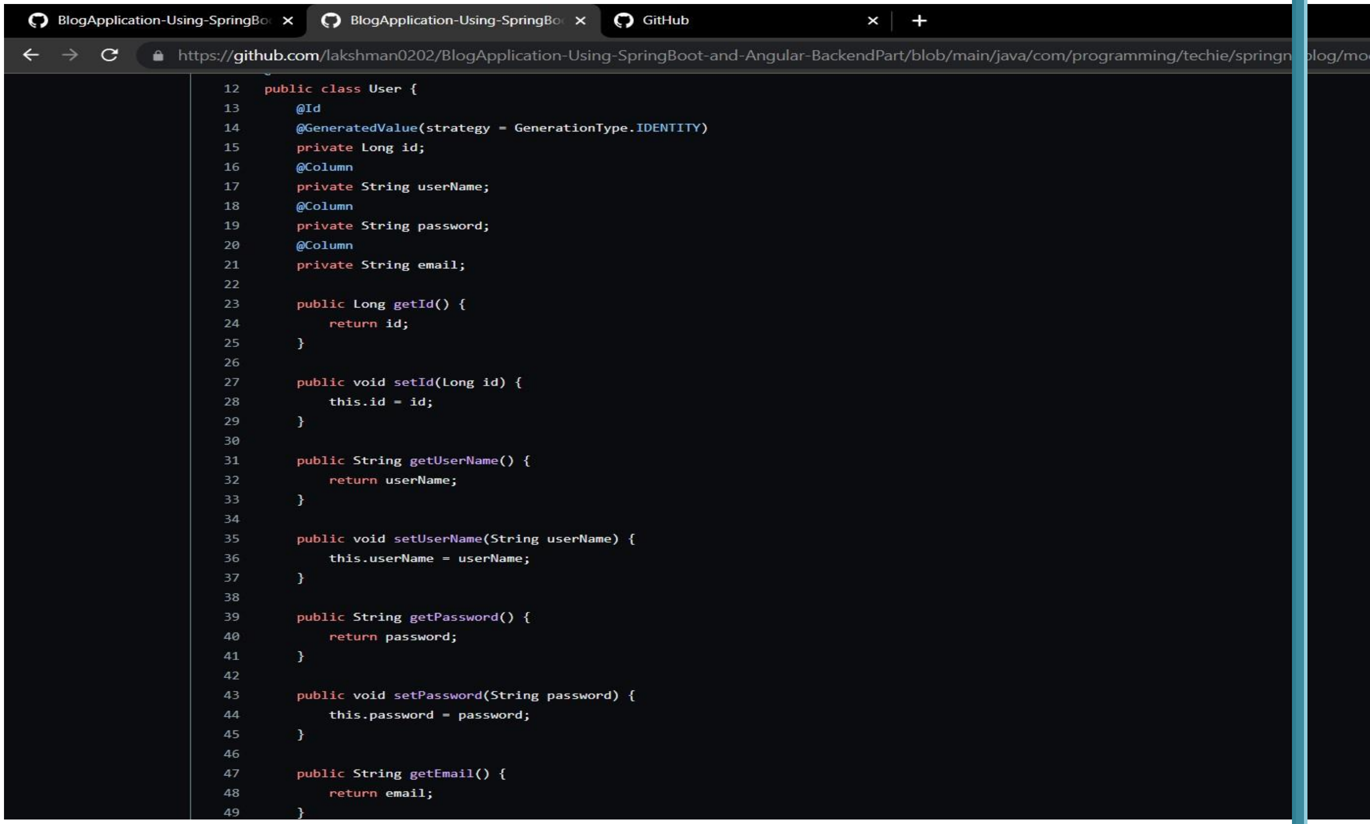

lakshman0202 Add files via upload

```java
package com.programming.techie.springngblog.exception;

@SuppressWarnings("serial")
public class PostNotFoundException extends RuntimeException {
    public PostNotFoundException(String message) {
        super(message);
    }
}
```

Post -


https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/model/Post.java

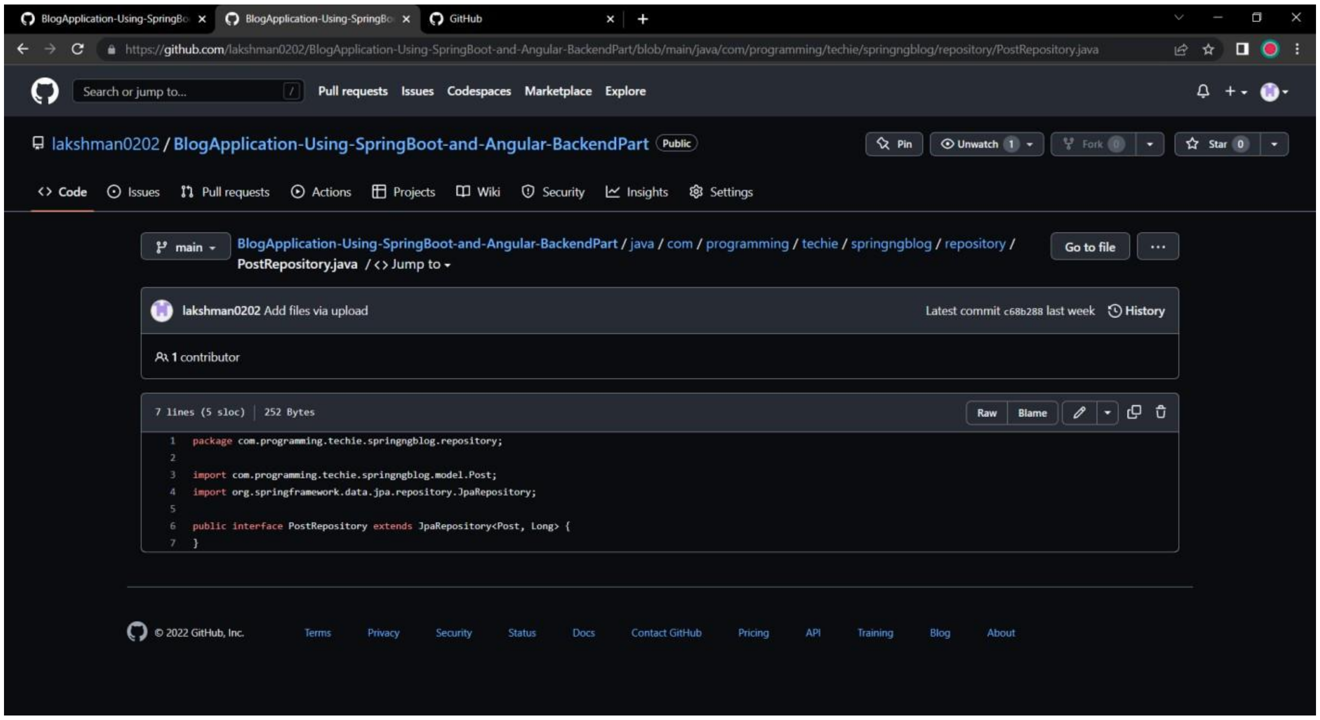
```java
@Entity
@Table
public class Post {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column
    private String title;
    @Lob
    @Column

    private String content;
    @Column
    private Instant createdOn;
    @Column
    private Instant updatedOn;
    @Column

    private String username;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getContent() {
```



BlogApplication-Using-SpringBo × BlogApplication-Using-SpringBo × GitHub ×

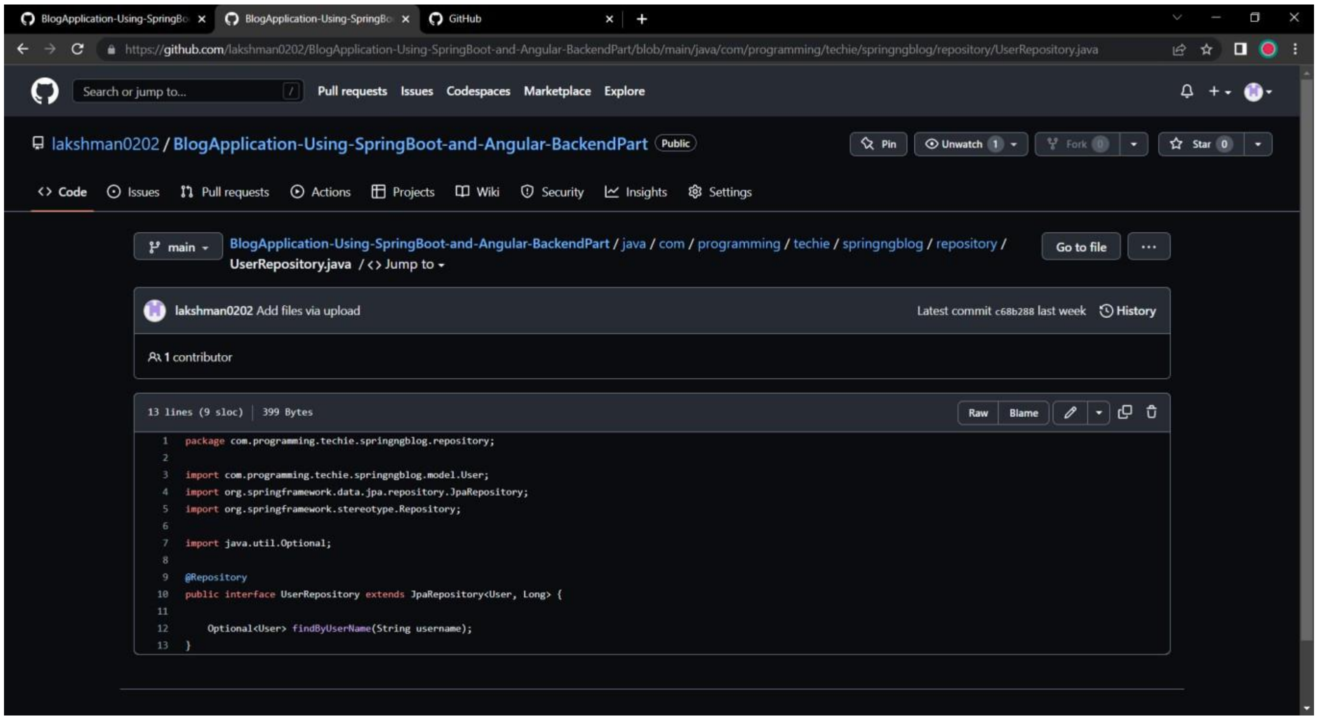
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springn blog/mo

```java
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column
    private String userName;
    @Column
    private String password;
    @Column
    private String email;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }
```


https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/repository/PostRepository.java
lakshman0202 / BlogApplication-Using-SpringBoot-and-Angular-BackendPart Public
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
main
BlogApplication-Using-SpringBoot-and-Angular-BackendPart / java / com / programming / techie / springngblog / repository / PostRepository.java
lakshman0202 Add files via upload
Latest commit c68b288 last week History
1 contributor
7 lines (5 sloc) 252 Bytes
Raw Blame
© 2022 GitHub, Inc.

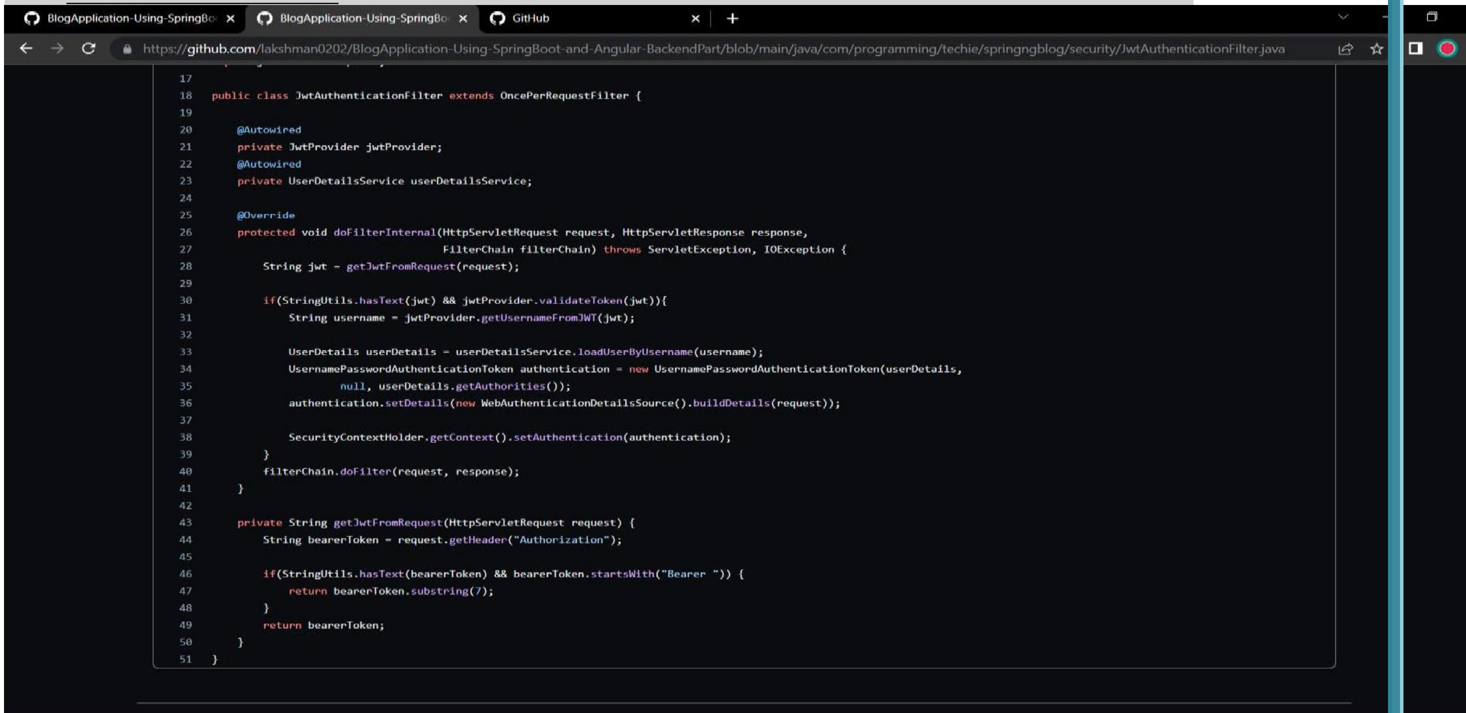
```java
package com.programming.techie.springngblog.repository;

import com.programming.techie.springngblog.model.Post;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PostRepository extends JpaRepository<Post, Long> {
}
```

Repository classes

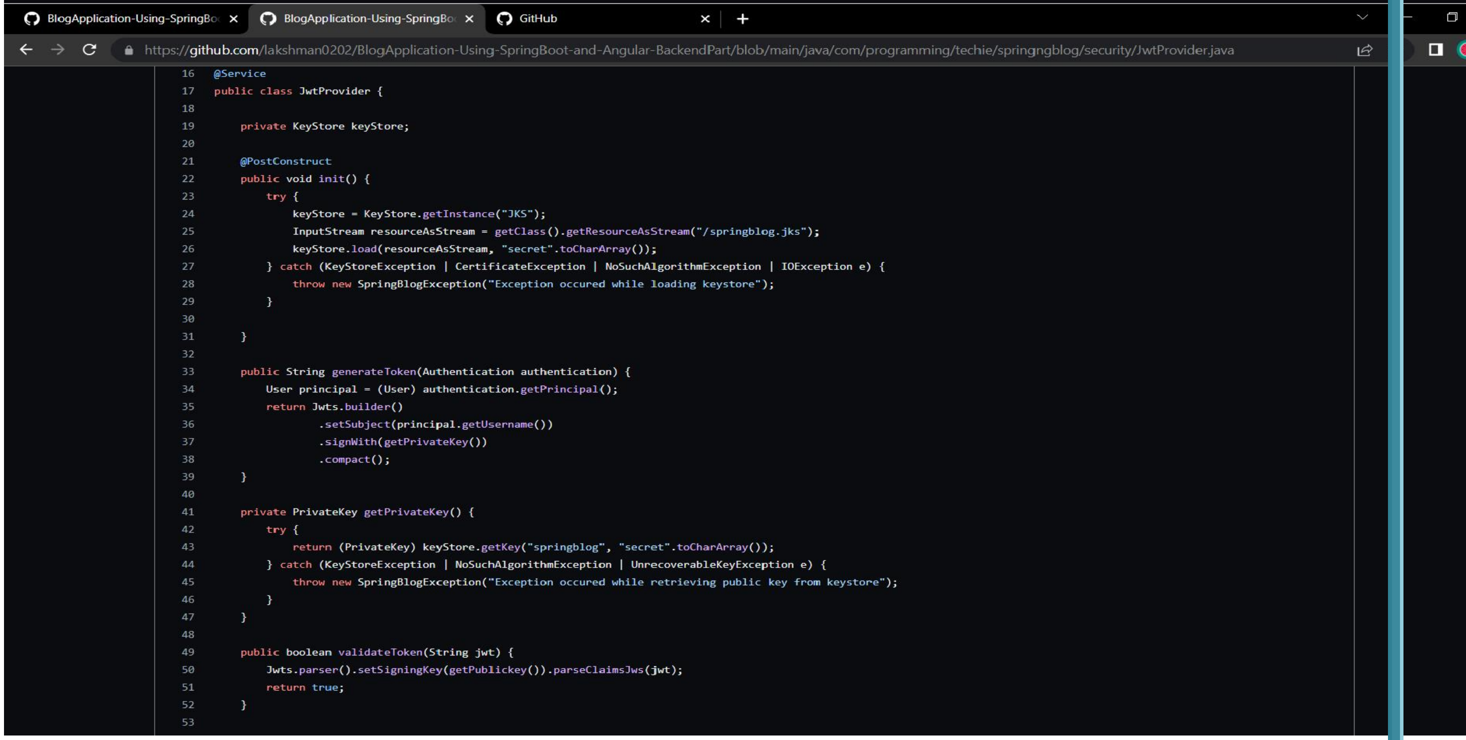

https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/repository/UserRepository.java
lakshman0202 / BlogApplication-Using-SpringBoot-and-Angular-BackendPart Public
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
main
BlogApplication-Using-SpringBoot-and-Angular-BackendPart / java / com / programming / techie / springngblog / repository / UserRepository.java
lakshman0202 Add files via upload
Latest commit c68b288 last week History
1 contributor
13 lines (9 sloc) 399 Bytes
Raw Blame

```java
package com.programming.techie.springngblog.repository;

import com.programming.techie.springngblog.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    Optional<User> findByUserName(String username);
}
```

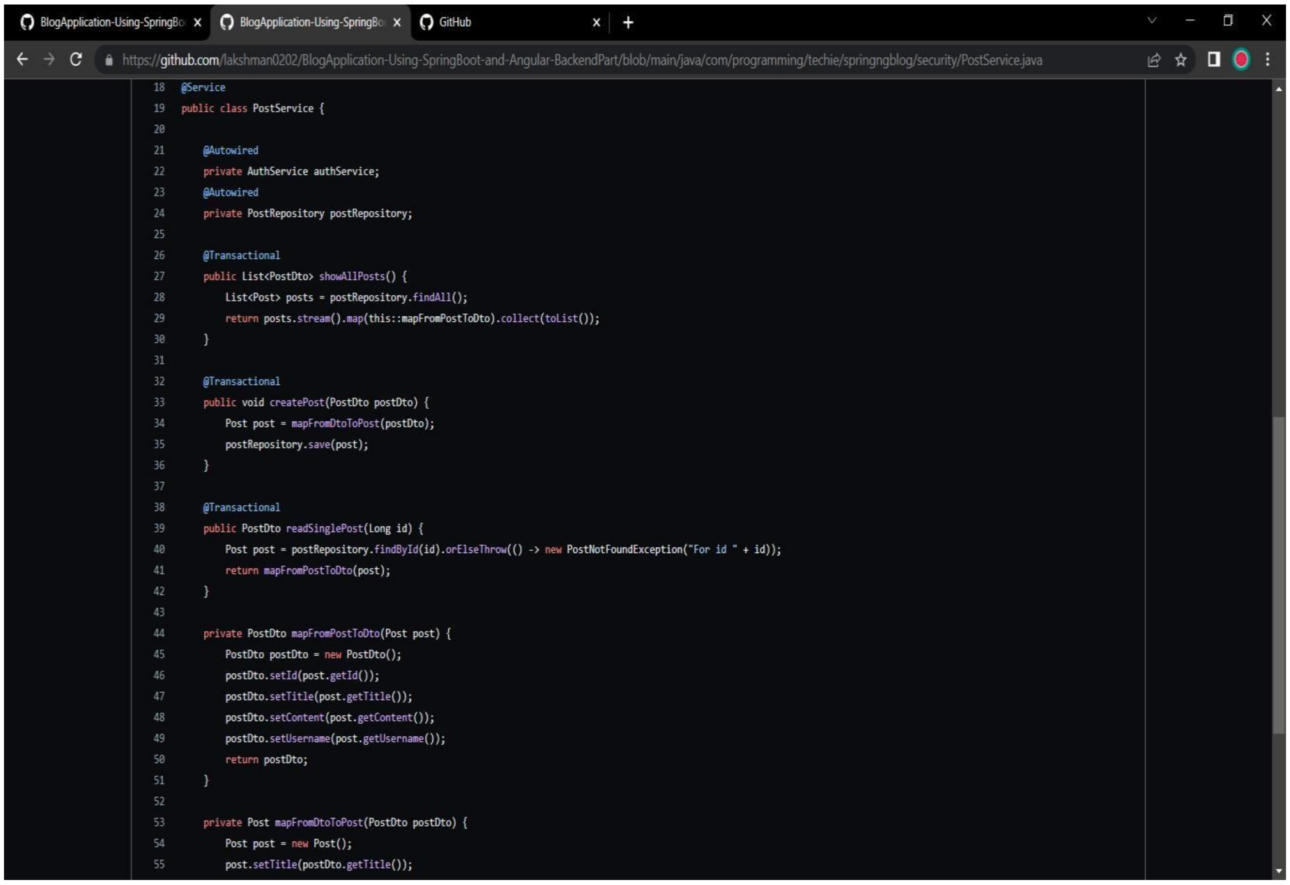
JWT


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/security/JwtAuthenticationFilter.java

```java

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired
    private JwtProvider jwtProvider;
    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {
        String jwt = getJwtFromRequest(request);

        if(StringUtils.hasText(jwt) && jwtProvider.validateToken(jwt)){
            String username = jwtProvider.getUsernameFromJWT(jwt);

            UserDetails userDetails = userDetailsService.loadUserByUsername(username);
            UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(userDetails,
                    null, userDetails.getAuthorities());
            authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        filterChain.doFilter(request, response);
    }

    private String getJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");

        if(StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring(7);
        }
        return bearerToken;
    }
}
```

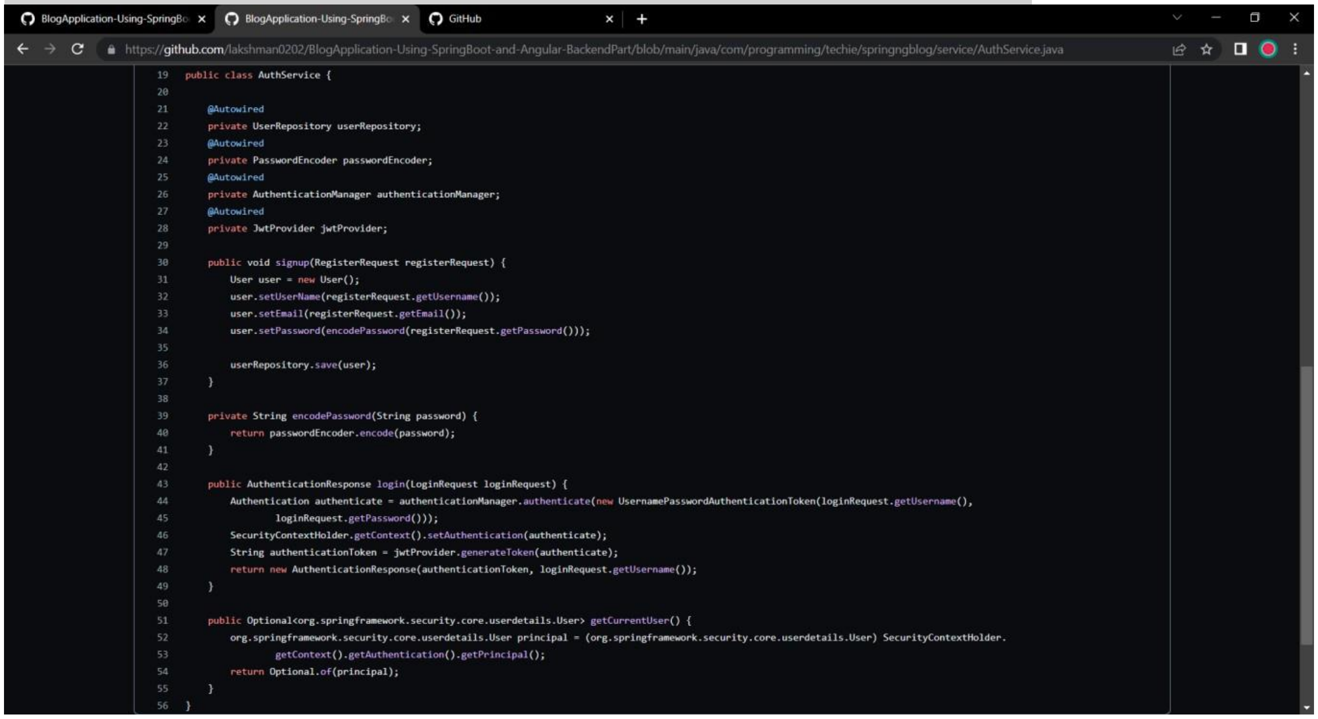

BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/security/JwtProvider.java

```java
@Service
public class JwtProvider {

    private KeyStore keyStore;

    @PostConstruct
    public void init() {
        try {
            keyStore = KeyStore.getInstance("JKS");
            InputStream resourceAsStream = getClass().getResourceAsStream("/springblog.jks");
            keyStore.load(resourceAsStream, "secret".toCharArray());
        } catch (KeyStoreException | CertificateException | NoSuchAlgorithmException | IOException e) {
            throw new SpringBlogException("Exception occured while loading keystore");
        }

    }

    public String generateToken(Authentication authentication) {
        User principal = (User) authentication.getPrincipal();
        return Jwts.builder()
                .setSubject(principal.getUsername())
                .signWith(getPrivateKey())
                .compact();
    }

    private PrivateKey getPrivateKey() {
        try {
            return (PrivateKey) keyStore.getKey("springblog", "secret".toCharArray());
        } catch (KeyStoreException | NoSuchAlgorithmException | UnrecoverableKeyException e) {
            throw new SpringBlogException("Exception occured while retrieving public key from keystore");
        }
    }

    public boolean validateToken(String jwt) {
        Jwts.parser().setSigningKey(getPublickey()).parseClaimsJws(jwt);
        return true;
    }

```


BlogApplication-Using-SpringBo
BlogApplication-Using-SpringBo
GitHub
https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/security/PostService.java

```java
@Service
public class PostService {

    @Autowired
    private AuthService authService;
    @Autowired
    private PostRepository postRepository;

    @Transactional
    public List<PostDto> showAllPosts() {
        List<Post> posts = postRepository.findAll();
        return posts.stream().map(this::mapFromPostToDto).collect(toList());
    }

    @Transactional
    public void createPost(PostDto postDto) {
        Post post = mapFromDtoToPost(postDto);
        postRepository.save(post);
    }

    @Transactional
    public PostDto readSinglePost(Long id) {
        Post post = postRepository.findById(id).orElseThrow(() -> new PostNotFoundException("For id " + id));
        return mapFromPostToDto(post);
    }

    private PostDto mapFromPostToDto(Post post) {
        PostDto postDto = new PostDto();
        postDto.setId(post.getId());
        postDto.setTitle(post.getTitle());
        postDto.setContent(post.getContent());
        postDto.setUsername(post.getUsername());
        return postDto;
    }

    private Post mapFromDtoToPost(PostDto postDto) {
        Post post = new Post();
        post.setTitle(postDto.getTitle());
```

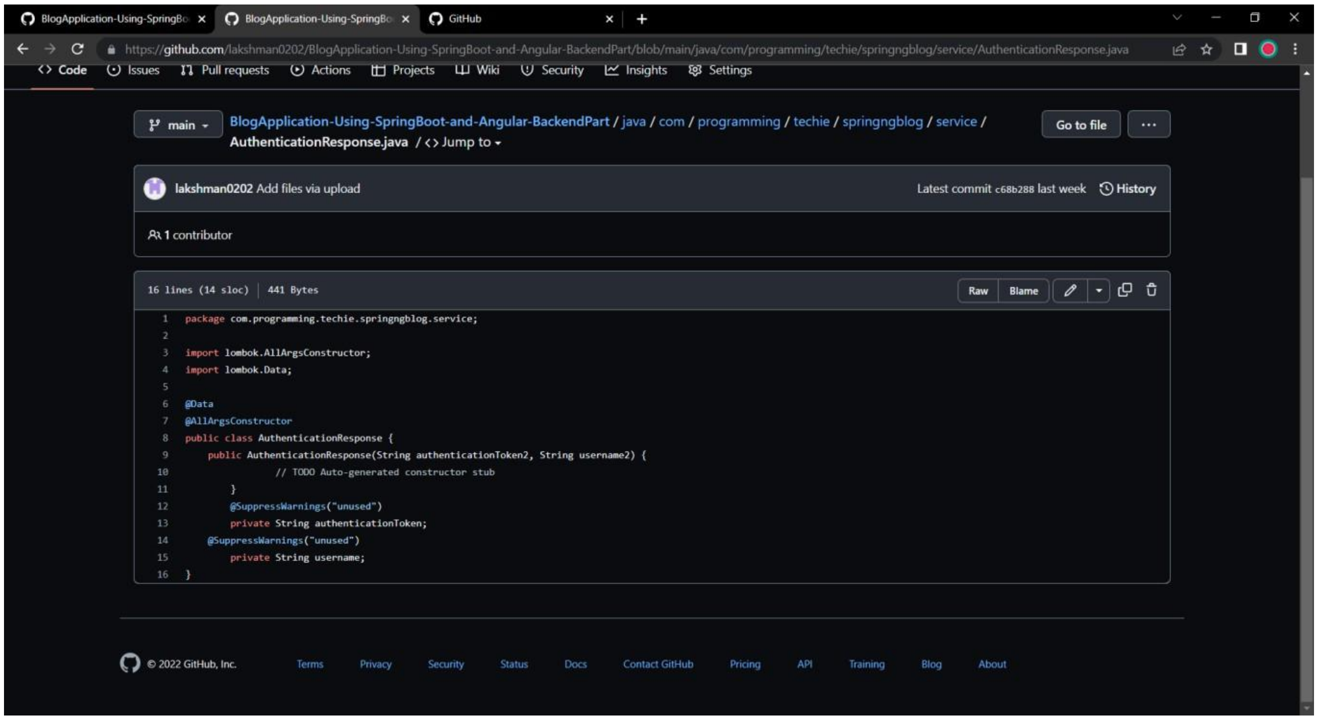
Auth service



https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/service/AuthService.java

```java
public class AuthService {

    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private AuthenticationManager authenticationManager;
    @Autowired
    private JwtProvider jwtProvider;

    public void signup(RegisterRequest registerRequest) {
        User user = new User();
        user.setUserName(registerRequest.getUsername());
        user.setEmail(registerRequest.getEmail());
        user.setPassword(encodePassword(registerRequest.getPassword()));

        userRepository.save(user);
    }

    private String encodePassword(String password) {
        return passwordEncoder.encode(password);
    }

    public AuthenticationResponse login(LoginRequest loginRequest) {
        Authentication authenticate = authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(loginRequest.getUsername(),
                loginRequest.getPassword()));
        SecurityContextHolder.getContext().setAuthentication(authenticate);
        String authenticationToken = jwtProvider.generateToken(authenticate);
        return new AuthenticationResponse(authenticationToken, loginRequest.getUsername());
    }

    public Optional<org.springframework.security.core.userdetails.User> getCurrentUser() {
        org.springframework.security.core.userdetails.User principal = (org.springframework.security.core.userdetails.User) SecurityContextHolder.
                getContext().getAuthentication().getPrincipal();
        return Optional.of(principal);
    }
}
```

Authentication response



https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/service/AuthenticationResponse.java

BlogApplication-Using-SpringBoot-and-Angular-BackendPart / java / com / programming / techie / springngblog / service / AuthenticationResponse.java

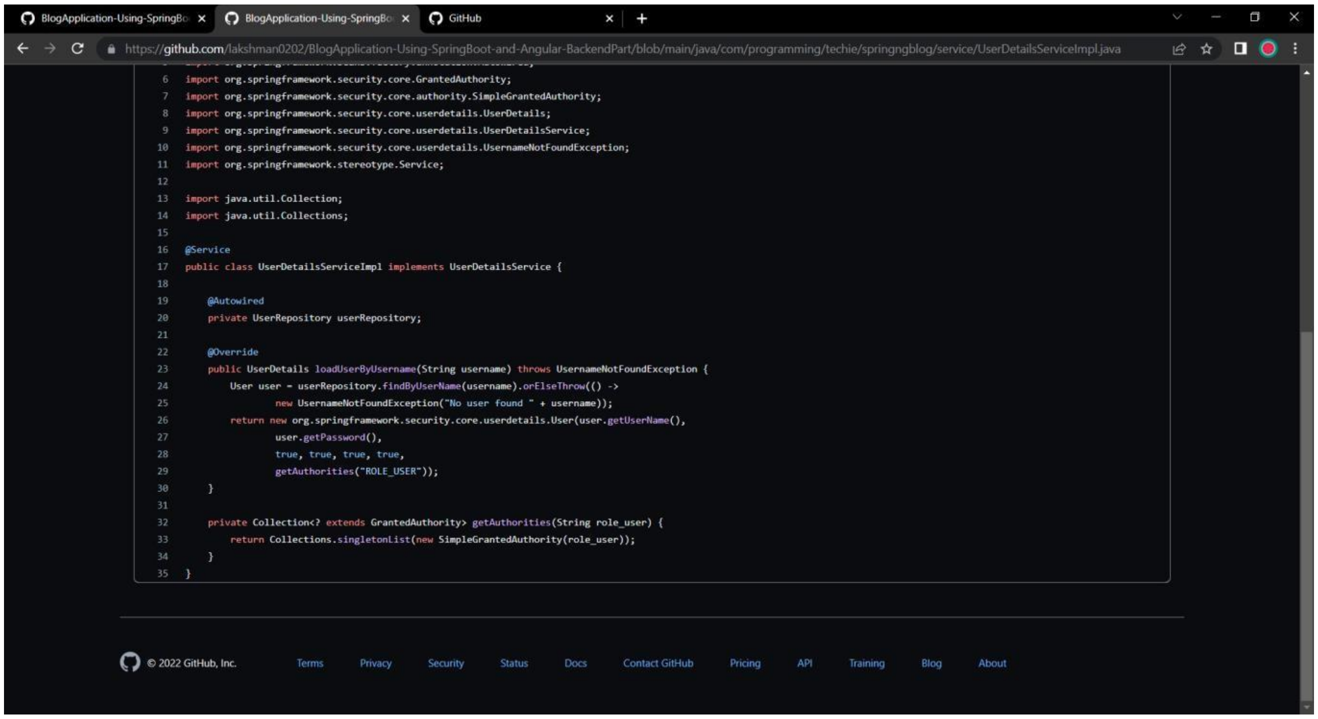
lakshman0202 Add files via upload — Latest commit c68b288 last week

1 contributor

16 lines (14 sloc) | 441 Bytes

```java
package com.programming.techie.springngblog.service;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class AuthenticationResponse {
    public AuthenticationResponse(String authenticationToken2, String username2) {
                // TODO Auto-generated constructor stub
        }
        @SuppressWarnings("unused")
        private String authenticationToken;
    @SuppressWarnings("unused")
        private String username;
}
```

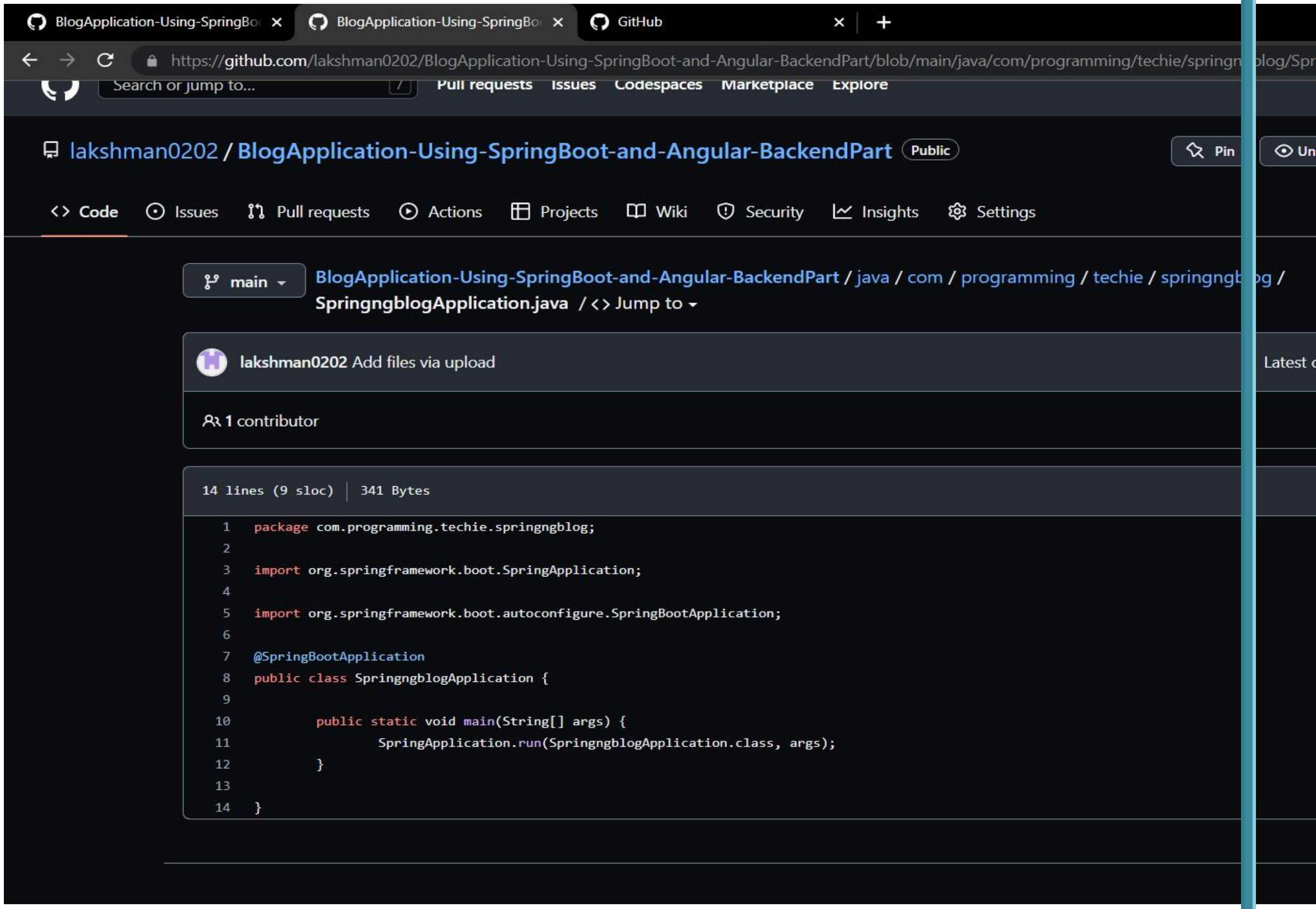
© 2022 GitHub, Inc.



BlogApplication-Using-SpringBo | BlogApplication-Using-SpringBo | GitHub

https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springngblog/service/UserDetailsServiceImpl.java

```java
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Collection;
import java.util.Collections;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUserName(username).orElseThrow(() ->
                new UsernameNotFoundException("No user found " + username));
        return new org.springframework.security.core.userdetails.User(user.getUserName(),
                user.getPassword(),
                true, true, true, true,
                getAuthorities("ROLE_USER"));
    }

    private Collection<? extends GrantedAuthority> getAuthorities(String role_user) {
        return Collections.singletonList(new SimpleGrantedAuthority(role_user));
    }
}
```


BlogApplication-Using-SpringBo | BlogApplication-Using-SpringBo | GitHub

https://github.com/lakshman0202/BlogApplication-Using-SpringBoot-and-Angular-BackendPart/blob/main/java/com/programming/techie/springn

lakshman0202 / BlogApplication-Using-SpringBoot-and-Angular-BackendPart Public

main BlogApplication-Using-SpringBoot-and-Angular-BackendPart / java / com / programming / techie / springngblog / SpringngblogApplication.java

lakshman0202 Add files via upload

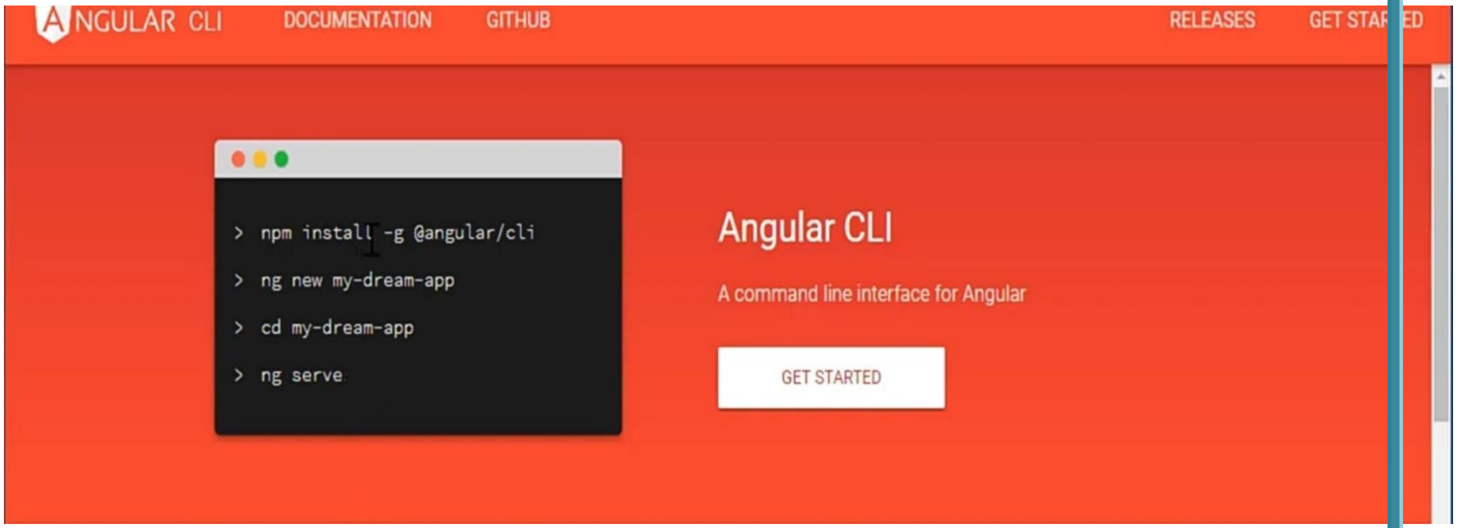
1 contributor

14 lines (9 sloc) | 341 Bytes

```
package com.programming.techie.springngblog;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringngblogApplication {

        public static void main(String[] args) {
                SpringApplication.run(SpringngblogApplication.class, args);
        }

}
```

FRONTEND PART USING ANGULAR

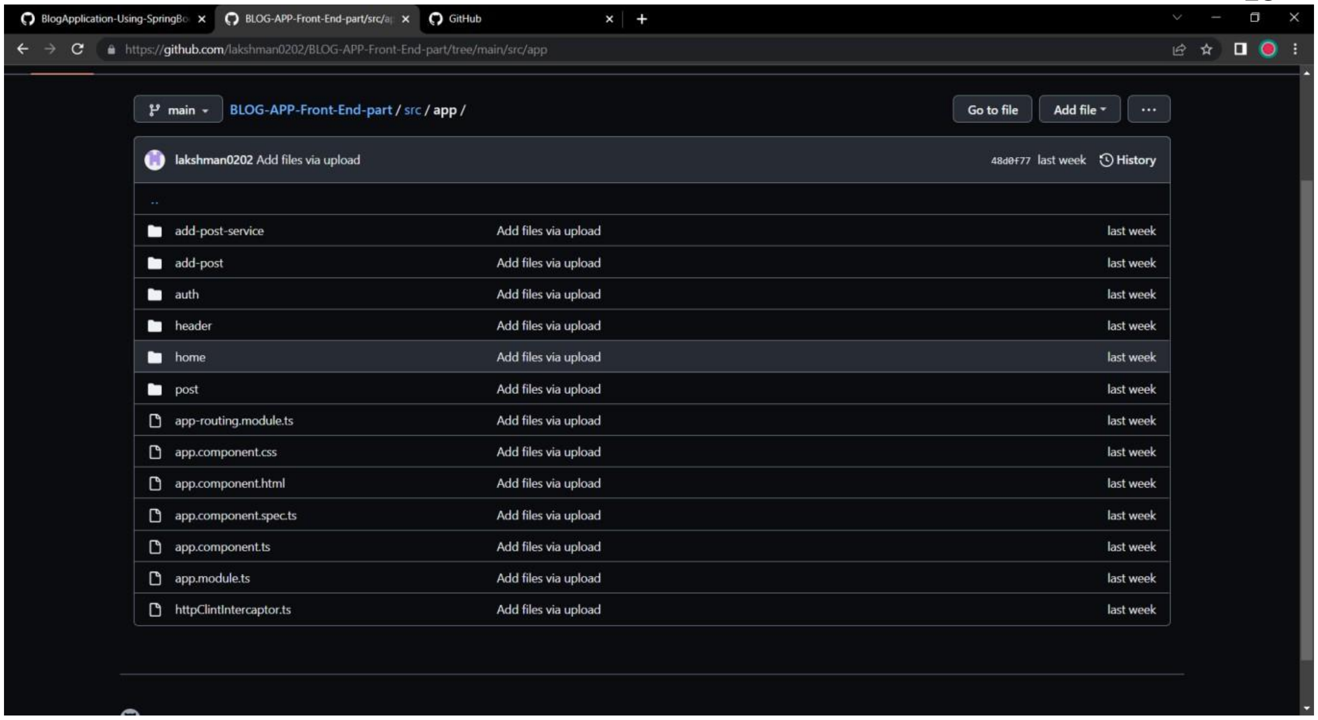
This the project structure for my front end part. I have implementing the technologies using HTML, CSS, Angular in this part .
Lets see the details Screenshots one by one ..I have attached in the pages below



```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Package structure

Lets see my structure of the project in the screenshot below



https://github.com/lakshman0202/BLOG-APP-Front-End-part/tree/main/src/app

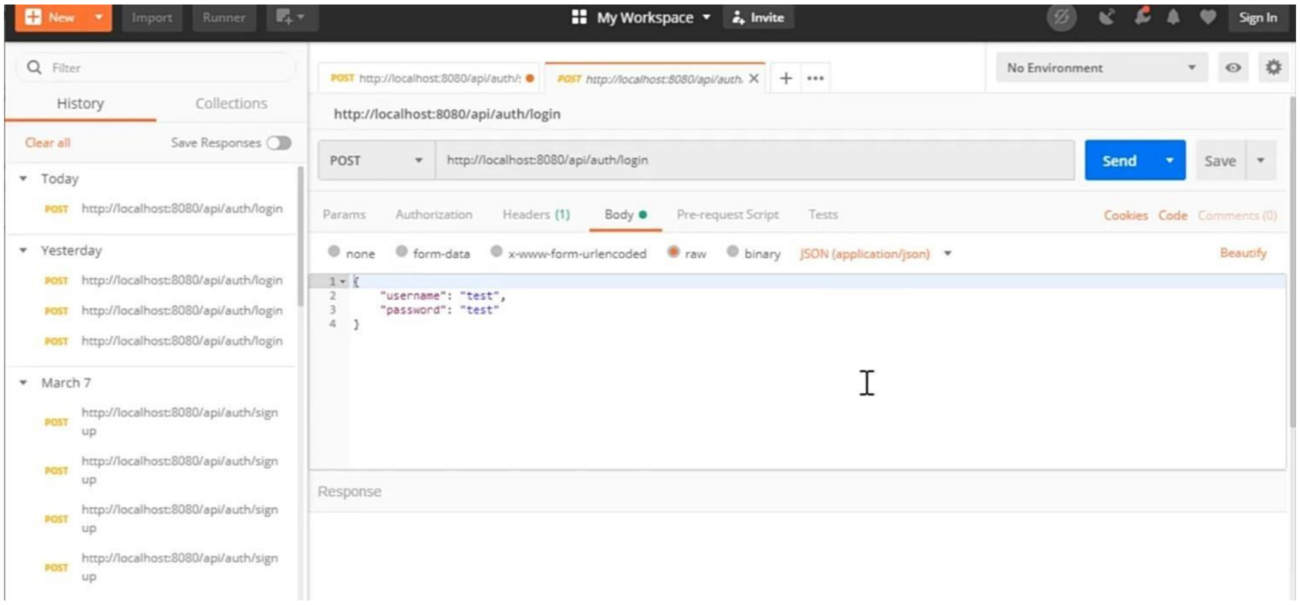
main BLOG-APP-Front-End-part / src / app /

lakshman0202 Add files via upload 48d0f77 last week History

..		
add-post-service	Add files via upload	last week
add-post	Add files via upload	last week
auth	Add files via upload	last week
header	Add files via upload	last week
home	Add files via upload	last week
post	Add files via upload	last week
app-routing.module.ts	Add files via upload	last week
app.component.css	Add files via upload	last week
app.component.html	Add files via upload	last week
app.component.spec.ts	Add files via upload	last week
app.component.ts	Add files via upload	last week
app.module.ts	Add files via upload	last week
httpClintIntercaptor.ts	Add files via upload	last week

Blog system screenshots

API Testing using postman



http://localhost:8080/api/auth/login

POST http://localhost:8080/api/auth/login

Body: raw JSON (application/json)

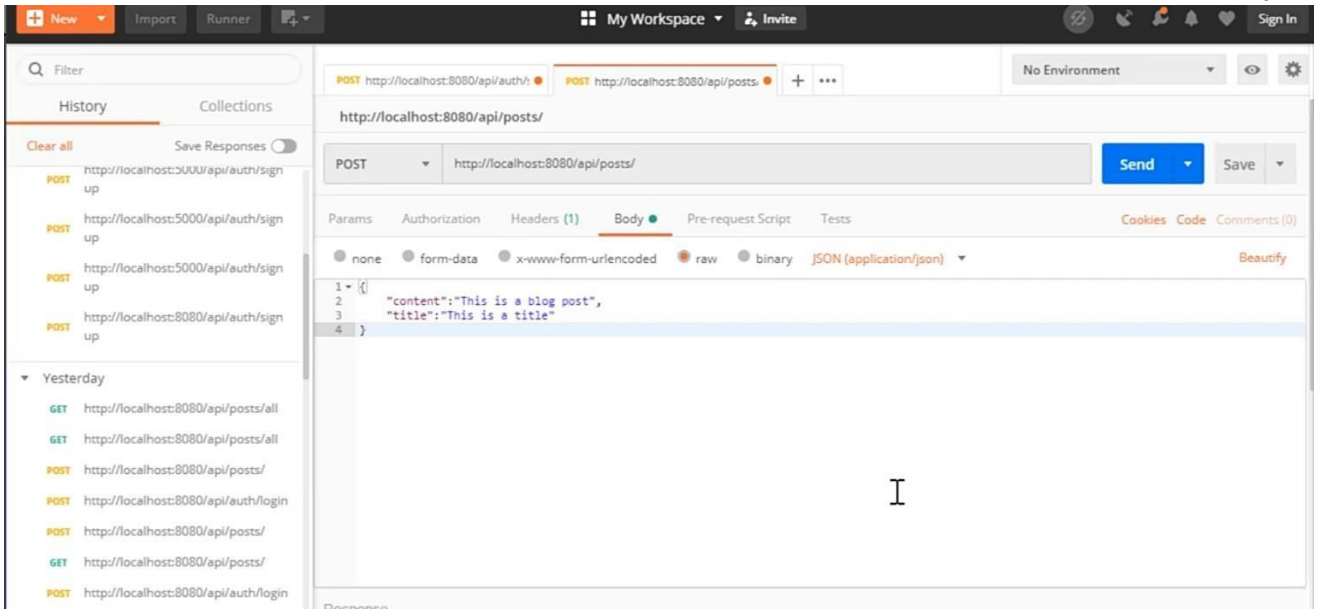
```
{
    "username": "test",
    "password": "test"
}
```

History:

Today
- POST http://localhost:8080/api/auth/login

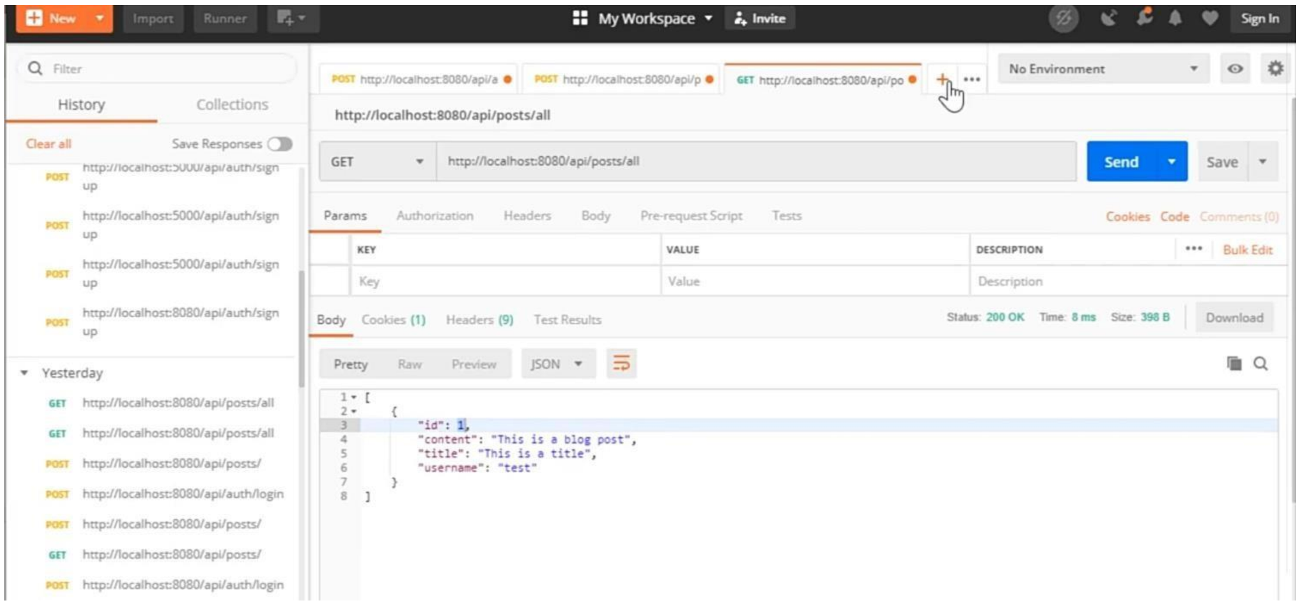
Yesterday
- POST http://localhost:8080/api/auth/login
- POST http://localhost:8080/api/auth/login
- POST http://localhost:8080/api/auth/login

March 7
- POST http://localhost:8080/api/auth/signup
- POST http://localhost:8080/api/auth/signup
- POST http://localhost:8080/api/auth/signup
- POST http://localhost:8080/api/auth/signup



POST http://localhost:8080/api/posts/

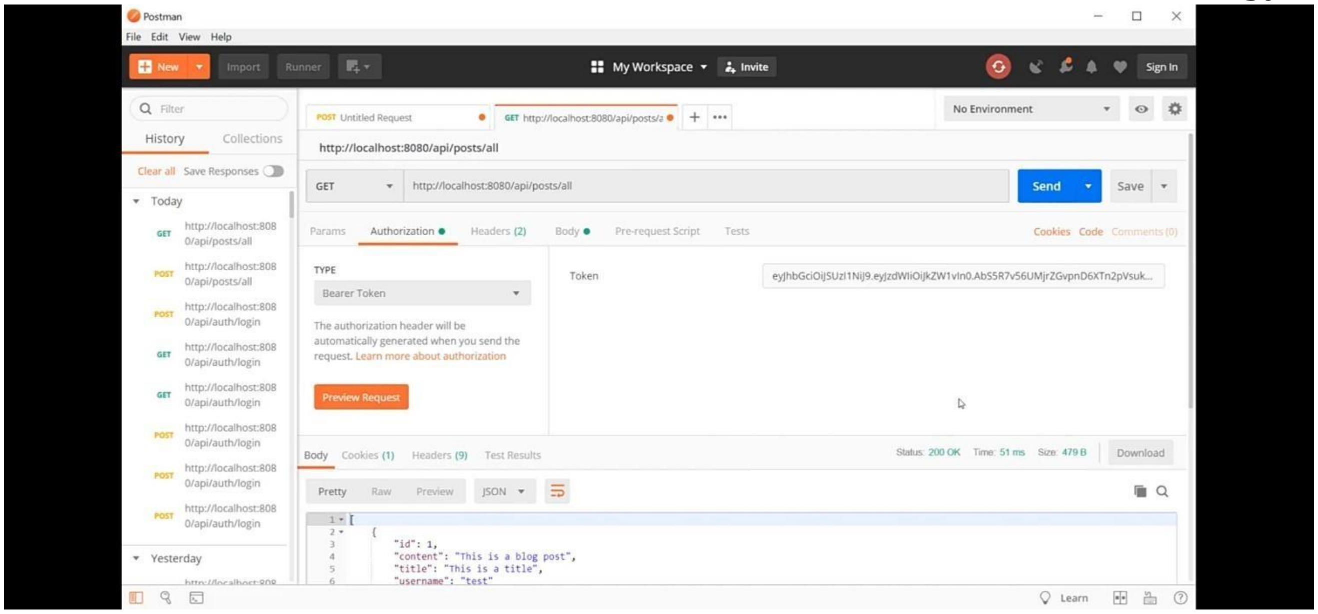
```
{
    "content":"This is a blog post",
    "title":"This is a title"
}
```



GET http://localhost:8080/api/posts/all

Status: 200 OK Time: 8 ms Size: 398 B

```
[
    {
        "id": 1,
        "content": "This is a blog post",
        "title": "This is a title",
        "username": "test"
    }
]
```



GET http://localhost:8080/api/posts/all

TYPE: Bearer Token

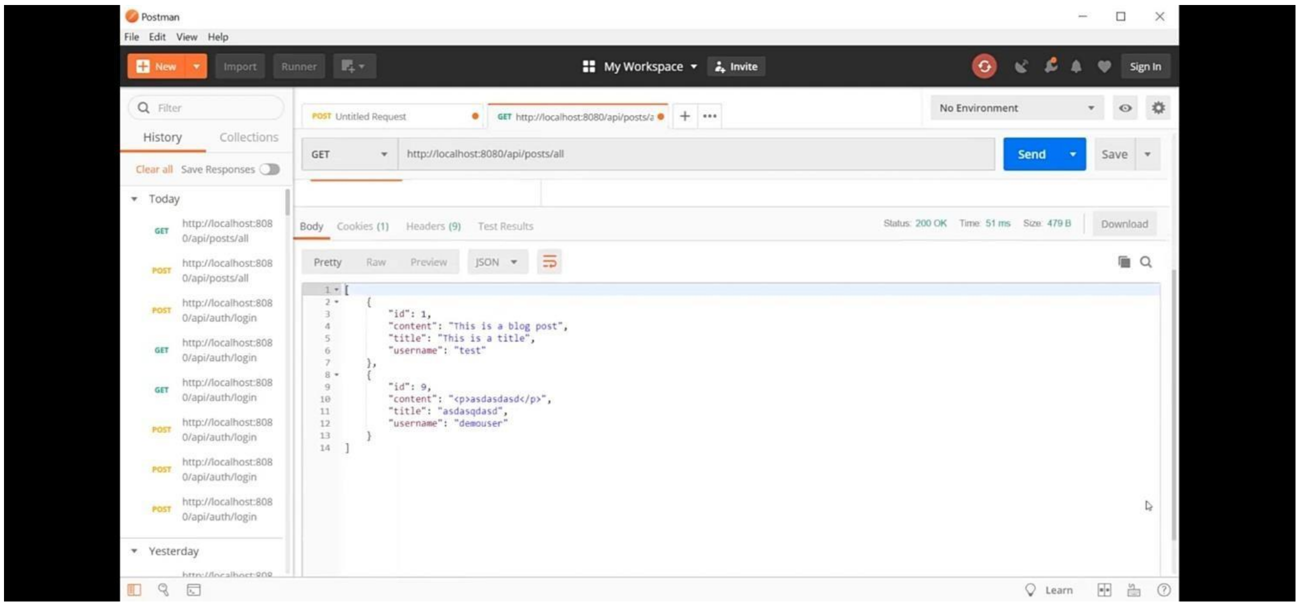
The authorization header will be automatically generated when you send the request. Learn more about authorization

Token: eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJkZW1vIn0.AbS5R7v56UMjrZGvpnD6XTn2pVsuk...

Status: 200 OK Time: 51 ms Size: 479 B

```
[
    {
        "id": 1,
        "content": "This is a blog post",
        "title": "This is a title",
        "username": "test"
```

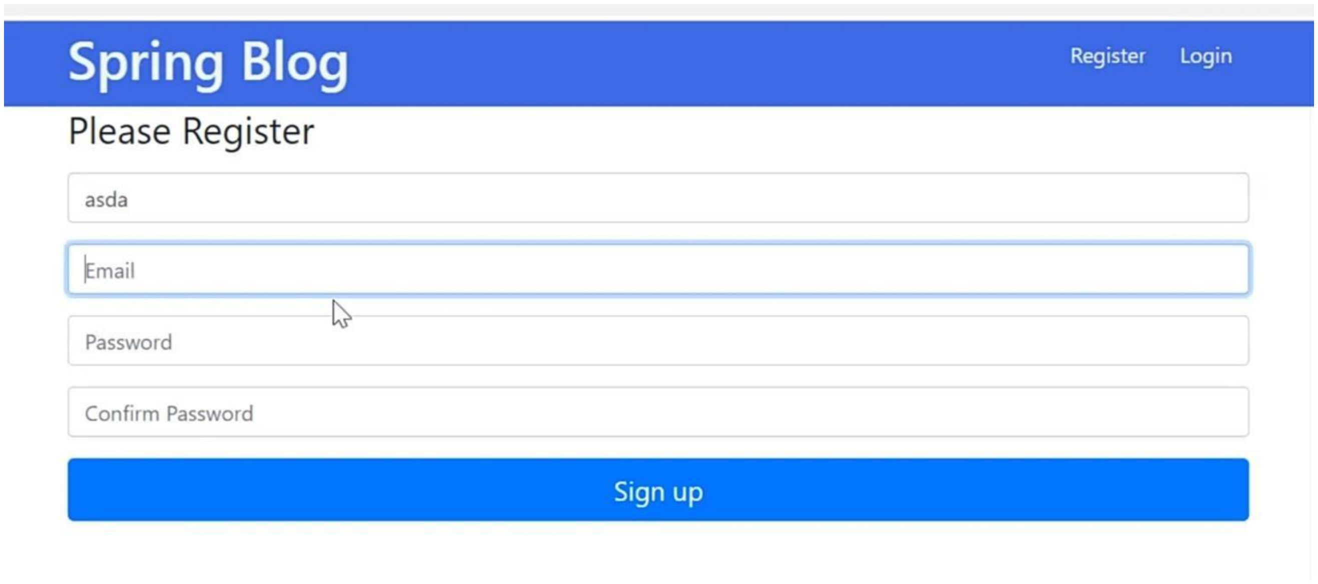
.



GET http://localhost:8080/api/posts/all

Status: 200 OK Time: 51 ms Size: 479 B

```
[
    {
        "id": 1,
        "content": "This is a blog post",
        "title": "This is a title",
        "username": "test"
    },
    {
        "id": 9,
        "content": "<p>asdasdasd</p>",
        "title": "asdasqdasd",
        "username": "demouser"
    }
]
```



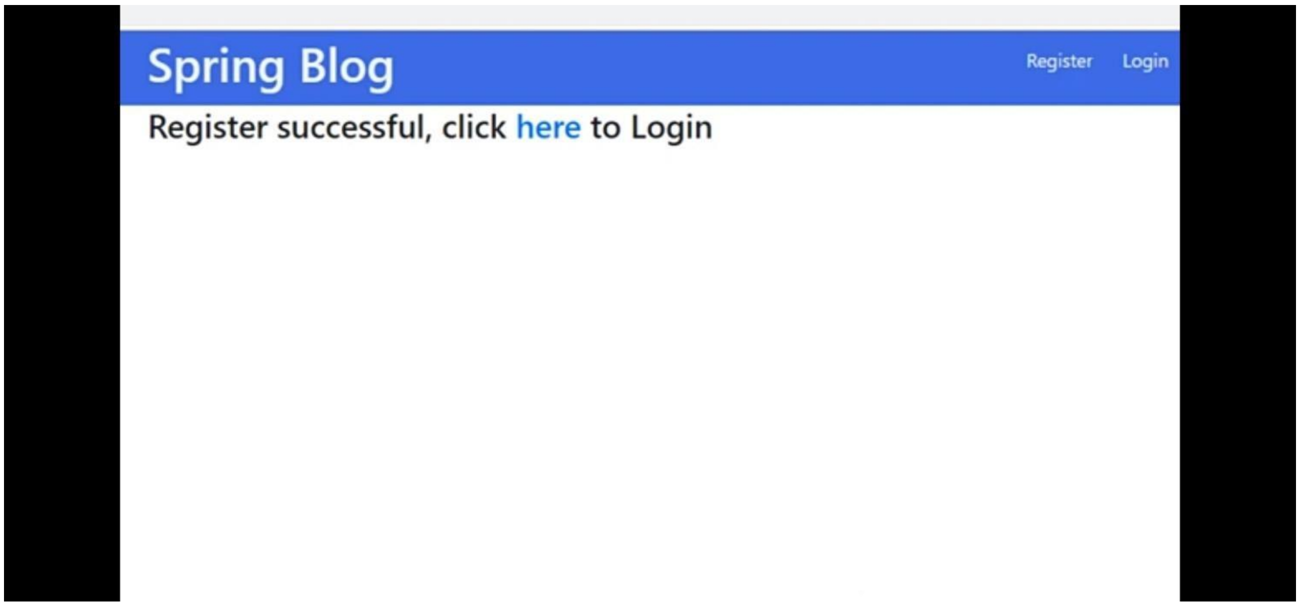
Please Register

asda

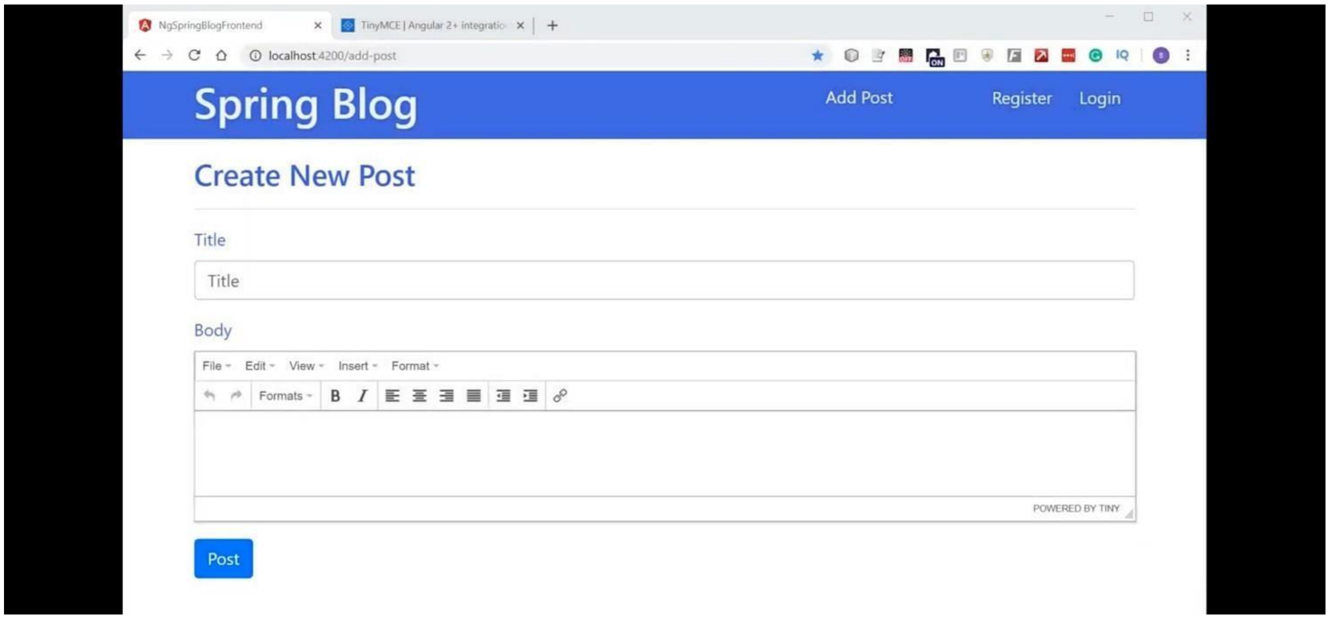
Email

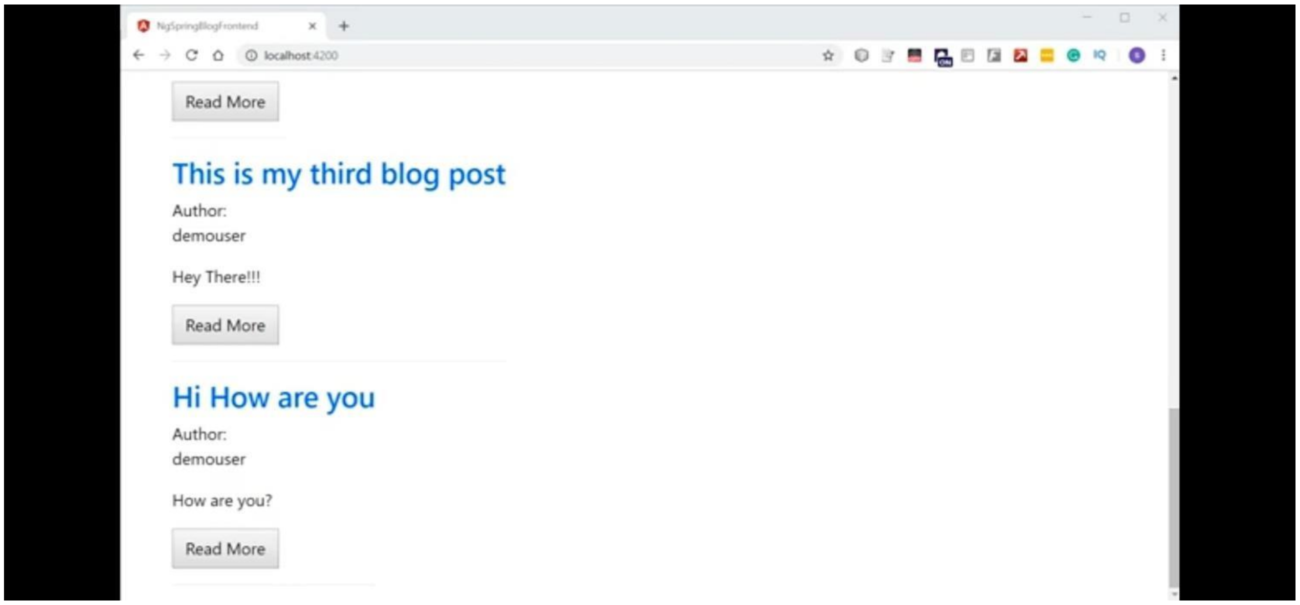
Password

Confirm Password



Register successful, click here to Login





This is my third blog post

Author:
demouser

Hey There!!!

Read More

Hi How are you

Author:
demouser

How are you?

Read More

Conclusion

Spring and Spring Boot are powerful tools in developing web applications. With the integrated Object Relational Mapping tools such as Hibernate web applications are developednot only faster but easier. Applications developed with Spring Boot and Hibernate are robust,loosely coupled and easy to use.

The objective of this project is to learn, understand the working of Spring Boot and Hibernate and implement a 'Personal Blog' integrating these two frameworks. The blog can beused to write and post articles, pictures and codes by the administrator. The blog also allows general users to read and comment on the passages.

.