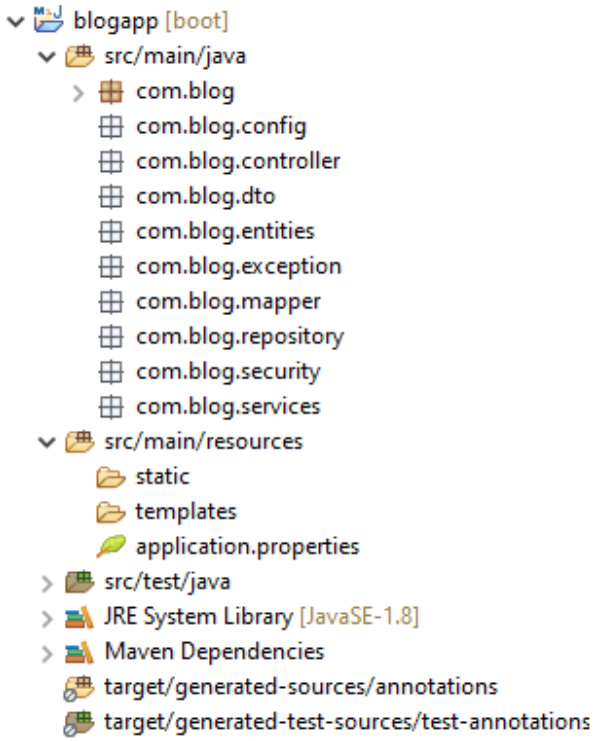
Thymeleaf Project development

Step 1: Create Project Structure



Step 2: Add following dependencies in pom.xml file

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>

        <parent>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>3.0.0-M3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>net.javaguides</groupId>
    <artifactId>springboot-blog-webapp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springboot-blog-webapp</name>
    <description>Spring Boot Thymeleaf Real-Time Web Application - Blog app</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
</dependency>
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
```

```
</dependencies>

<build>
	<plugins>
		<plugin>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-maven-plugin</artifactId>
			<configuration>
				<excludes>
					<exclude>

<groupId>org.projectlombok</groupId>
					<artifactId>lombok</artifactId>
					</exclude>
				</excludes>
			</configuration>
		</plugin>
	</plugins>
</build>
<repositories>
	<repository>
		<id>spring-milestones</id>
```

```
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>

</project>
```

Step 3: Add following details in application.properties file

```
#jdbc driver class name
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/myblogapp?
useSSL=false
spring.datasource.username=root
spring.datasource.password=test

spring.jpa.properties.dialect.hibernate=org.hibernate.dialec
t.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Step 4: Create Entity class

```
package com.blogapp.entities;

import java.time.LocalDateTime;

import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Lob;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
```

```java
@Builder
@Entity
@Table(name = "posts")
public class Post {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;

        @Column(nullable = false)
        private String title;
        private String url;

        @Lob
        @Column(nullable = false)
        private String content;
        private String shortDescription;

        @CreationTimestamp
        private LocalDateTime createdOn;

        @UpdateTimestamp
        private LocalDateTime updatedOn;

}
```

Step 5: Create Repository layer

```java
package com.blog.repository;

import com.blog.entities.Post;
import
org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

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

```
    Optional<Post> findByUrl(String url);
}
```

Step 6: Create DTO Class

```
package com.blog.dto;


import jakarta.validation.constraints.NotEmpty;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PostDto {
    private Long id;
    @NotEmpty(message = "Post title should not be empty")
    private String title;
    private String url;
    @NotEmpty(message = "Post content should not be empty")
    private String content;
    @NotEmpty(message = "Post short description should be
empty")
    private String shortDescription;
    private LocalDateTime createdOn;
    private LocalDateTime updatedOn;
}
```

Step 7: Create Mapper Class

```
package com.blog.mapper;

import com.blog.dto.PostDto;
import com.blog.entities.Post;

public class PostMapper {
    // map Post entity to PostDto
    public static PostDto mapToPostDto(Post post){
        return PostDto.builder()
                .id(post.getId())
                .title(post.getTitle())
                .url(post.getUrl())
                .content(post.getContent())
                .shortDescription(post.getShortDescription())
                .createdOn(post.getCreatedOn())
                .updatedOn(post.getUpdatedOn())
                .build();
    }

    // map Postdto to Post entity
    public static Post mapToPost(PostDto postDto){
        return Post.builder()
                .id(postDto.getId())
                .title(postDto.getTitle())
                .content(postDto.getContent())
                .url(postDto.getUrl())
                .shortDescription(postDto.getShortDescription())
                .createdOn(postDto.getCreatedOn())
                .updatedOn(postDto.getUpdatedOn())
                .build();
    }
}
```

Creating List All Post Feature

Step 1: Create Service Layer

Create PostService Interface:

```
package com.blog.services;

import com.blog.dto.PostDto;
```

```
import java.util.List;

public interface PostService {
    List<PostDto> findAllPosts();
}
```

Create PostServiceImpl class

```
package com.blog.services.impl;

import com.blog.dto.PostDto;
import com.blog.entities.Post;
import com.blog.mapper.PostMapper;
import com.blog.repository.PostRepository;
import com.blog.services.PostService;

import java.util.List;
import java.util.stream.Collectors;

public class PostServiceImpl implements PostService {

    private PostRepository postRepository;

    public PostServiceImpl(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    @Override
    public List<PostDto> findAllPosts() {
        List<Post> posts = postRepository.findAll();

        return posts.stream().map(PostMapper::mapToPostDto)
                .collect(Collectors.toList());
    }
}
```

Step 2: Create Controller Class:

```
package com.blog.controller;

import com.blog.dto.PostDto;
import com.blog.services.PostService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;
```

```java
@Controller
public class PostController {

    private PostService postService;

    public PostController(PostService postService) {
        this.postService = postService;
    }

    // create handler method, GET request and return model and view
    @GetMapping("/admin/posts")
    public String posts(Model model){
        List<PostDto> posts = postService.findAllPosts();
        model.addAttribute("posts", posts);
        return "/admin/posts";
    }
}
```

Step 4: Add data to tables using following script

INSERT INTO `posts` VALUES

(1,' Content goes here','2022-07-18 10:45:18.617432','In this blog post, you will important OOPS concepts in Java with examples','OOPS Concepts in Java','2022-07-18 10:45:18.617561','oops-concepts-in-java')

(2,' Content goes here','2022-07-18 10:45:18.630278','In this blog post, you will learn about Variables in Java with examples','Variables in Java','2022-07-18 10:45:18.630297','variables-in-java'),

(3,' Content goes here','2022-07-18 10:45:18.632620','In this blog post, you will learn about Primitive Data Types in Java with examples','Primitive Data Types in Java','2022-07-18 10:45:18.632632','primitive-data-types-in-java'),

(4,' Content goes here','2022-07-18 10:45:18.634347','In this blog post, you will learn about Access Modifiers in Java with examples','Access Modifiers in Java','2022-07-18 10:45:18.634357','access-modifiers-in-java'),

(5,' Content goes here','2022-07-18 10:45:18.635878','In this blog post, you will learn about Arrays in Java with examples','Arrays in Java','2022-07-18 10:45:18.635889','arrys-in-java');

Step 5: Create footer, header and posts.html thymeleaf templates

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>navbar</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark"
th:fragment="navbar">
  <a class="navbar-brand" href="#">Blog App</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-
expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavDropdown">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" th:href="@{/admin/posts}">Posts<span
class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">New Post</a>
      </li>

    </ul>
  </div>
</nav>
</body>
</html>
```

Navbar.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
  <!-- CSS only -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.
css" rel="stylesheet" integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">
</head>
<body>

</body>
</html>

<footer class="footer-copyright text-center bg-light py-3"
               style="position:fixed; width:100%; bottom:0"
               th:fragment="footer">
    <div>
        <strong>
            Copyright &copy; 2022 <a href="">Pankaj Sir Academy</a>
            All rights reserved.
        </strong>
    </div>
</footer>
```

Footer.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
```

```
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
                <td th:text="${post.createdOn}">post created date</td>
                <td></td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Posts.html

Implementing Create Post Feature

Step 1: Create handler method in controller layer

```
// handler method to handle new post request
@GetMapping("admin/posts/newpost")
public String newPostForm(Model model){
    PostDto postDto = new PostDto();
    model.addAttribute("post", postDto);
    return "admin/create_post";
}
```

PostController.java

Step 2: Create Thymeleaf view page

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>navbar</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

```
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark"
th:fragment="navbar">
  <a class="navbar-brand" href="#">Blog App</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-
expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavDropdown">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" th:href="@{/admin/posts}">Posts<span
class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">New Post</a>
      </li>

    </ul>
  </div>
</nav>
</body>
</html>
```

Navbar.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
  <!-- CSS only -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.
css" rel="stylesheet" integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">
</head>
<body>

</body>
</html>

<footer class="footer-copyright text-center bg-light py-3"
                style="position:fixed; width:100%; bottom:0"
                th:fragment="footer">
    <div>
        <strong>
            Copyright &copy; 2022 <a href="">Pankaj Sir Academy</a>
```

```
            All rights reserved.
        </strong>
    </div>
</footer>
```

Footer.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div th:replace="admin/navbar :: navbar"></div>
<div class="container">
  <br/>
  <br/>
<div class="card" >
  <form method="post" th:action="@{/admin/posts}" th:object="${post}">
    <div class="form-group card-body" >
      <label >Title</label>
      <input type="text" class="form-control" id="title"
placeholder="Enter Title" name="title" th:field="*{title}">
    </div>
    <div class="form-group card-body">
      <label >Title</label>
      <input type="text" class="form-control" id="shortDescription"
placeholder="Enter Short Description" name="shortDescription"
th:field="*{shortDescription}">
    </div>

    <div class="form-group card-body">
      <label >Content</label>
      <textarea class="form-control" id="content" rows="10" name="content"
th:field="*{content}"></textarea>
    </div>
    <div class="form-group card-body">
      <button type="submit" class="btn btn-primary card-
body">Submit</button>
    </div>

  </form>

</div>
  <br/>
```

```
  <br/>
  <br/>
  <br/>
<div th:replace="admin/footer :: footer"></div>
</div>
</body>
</html>
```

Create_post.html

Step 3: Create handler method in form to receive create_post form data

```
// handler method to handle form submit request
@PostMapping("/admin/posts")
public String createPost(@ModelAttribute("post") PostDto postDto,
                                                  Model model){
    postDto.setUrl(getUrl(postDto.getTitle()));
    postService.createPost(postDto);
    return "redirect:/admin/posts";
}

private static String getUrl(String postTitle){
    // OOPS Concepts Explained in Java
    // oops-concepts-explained-in-java
    String title = postTitle.trim().toLowerCase();
    String url = title.replaceAll("\\s+", "-");
    url = url.replaceAll("[^A-Za-z0-9]", "-");
    return url;
}
```

Step 4: Create Services method:

```
void createPost(PostDto postDto);
```

PostService.java

```
@Override
public void createPost(PostDto postDto) {
    Post post = PostMapper.mapToPost(postDto);
    postRepository.save(post);
}
```

PostServiceImpl.java

Steps to handle form validations

Step 1: Add validation jars from pom.xml file

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

Step 2: Add Following annotations in PostDto Class

```
package com.blog.dto;


import jakarta.validation.constraints.NotEmpty;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PostDto {
    private Long id;
    @NotEmpty(message = "Post title should not be empty")
    private String title;
    private String url;
    @NotEmpty(message = "Post content should not be empty")
    private String content;
    @NotEmpty(message = "Post short description should be empty")
    private String shortDescription;
    private LocalDateTime createdOn;
    private LocalDateTime updatedOn;
}
```

Step 3: Add Following Classes & code in controller Layer

```
1. BindingResult result

2. if(result.hasErrors()){
       model.addAttribute("post", postDto);
```

```
        return "admin/create_post";
    }
```

Example:

```
// handler method to handle form submit request
@PostMapping("/admin/posts")
public String createPost(@Valid @ModelAttribute("post") PostDto postDto,
                         BindingResult result,
                         Model model){
    if(result.hasErrors()){
        model.addAttribute("post", postDto);
        return "admin/create_post";
    }
    postDto.setUrl(getUrl(postDto.getTitle()));
    postService.createPost(postDto);
    return "redirect:/admin/posts";
}

private static String getUrl(String postTitle){
    // OOPS Concepts Explained in Java
    // oops-concepts-explained-in-java
    String title = postTitle.trim().toLowerCase();
    String url = title.replaceAll("\\s+", "-");
    url = url.replaceAll("[^A-Za-z0-9]", "-");
    return url;
}
```

Step 4: Add Following Code in create_posts.html file

```
<p th:if="${#fields.hasErrors('title')}" class="text-danger"
   th:errors="*{title}"></p>
```

Example:

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
```

```
crossorigin="anonymous">
</head>
<body>
<div th:replace="admin/navbar :: navbar"></div>
<div class="container">
  <br/>
  <br/>
<div class="card" >
  <form method="post" th:action="@{/admin/posts}" th:object="${post}">
    <div class="form-group card-body" >
      <label >Title</label>
      <input type="text" class="form-control" id="title"
placeholder="Enter Title" name="title" th:field="*{title}">
    </div>

    <p th:if="${#fields.hasErrors('title')}" class="text-danger"
       th:errors="*{title}"></p>

    <div class="form-group card-body">
      <label >Title</label>
      <input type="text" class="form-control" id="shortDescription"
placeholder="Enter Short Description" name="shortDescription"
th:field="*{shortDescription}">
    </div>

    <p th:if="${#fields.hasErrors('shortDescription')}" class="text-
danger"
       th:errors="*{shortDescription}"></p>

    <div class="form-group card-body">
      <label >Content</label>
      <textarea class="form-control" id="content" rows="10" name="content"
th:field="*{content}"></textarea>
    </div>

    <p th:if="${#fields.hasErrors('content')}" class="text-danger"
       th:errors="*{content}"></p>

    <div class="form-group card-body">
      <button type="submit" class="btn btn-primary card-
body">Submit</button>
    </div>

  </form>

</div>
  <br/>
  <br/>
  <br/>
  <br/>
<div th:replace="admin/footer :: footer"></div>
</div>
</body>
</html>
```

Adding CKEditor to your project:

Step 1: Add Following scripts to create_post.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div th:replace="admin/navbar :: navbar"></div>
<div class="container">
  <br/>
  <br/>
<div class="card" >
  <form method="post" th:action="@{/admin/posts}" th:object="${post}">
    <div class="form-group card-body" >
      <label >Title</label>
      <input type="text" class="form-control" id="title"
placeholder="Enter Title" name="title" th:field="*{title}">
    </div>
    <p th:if="${#fields.hasErrors('title')}" class="text-danger"
       th:errors="*{title}"></p>
    <div class="form-group card-body">
      <label >Title</label>
      <input type="text" class="form-control" id="shortDescription"
placeholder="Enter Short Description" name="shortDescription"
th:field="*{shortDescription}">
    </div>
    <p th:if="${#fields.hasErrors('shortDescription')}" class="text-
danger"
       th:errors="*{shortDescription}"></p>
    <div class="form-group card-body">
      <label >Content</label>
      <textarea class="form-control" id="content" rows="10" name="content"
th:field="*{content}"></textarea>
    </div>
    <p th:if="${#fields.hasErrors('content')}" class="text-danger"
       th:errors="*{content}"></p>
    <div class="form-group card-body">
      <button type="submit" class="btn btn-primary card-
body">Submit</button>
    </div>

  </form>
```

```
</div>
  <br/>
  <br/>
  <br/>
  <br/>
<div th:replace="admin/footer :: footer"></div>
</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"
        integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
        crossorigin="anonymous"></script>
<script
src="https://cdn.ckeditor.com/ckeditor5/34.2.0/classic/ckeditor.js"></scri
pt>
<script>
                        ClassicEditor
                                .create( document.querySelector(
'#content' ) )
                                .then( editor => {
                                        console.log( editor );
                                } )
                                .catch( error => {
                                        console.error( error );
                                } );
</script>


</body>
</html>
```

Create_post.html

Implementing Edit Feature for Admin User

Step 1: Adding Edit button to posts.html page

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
```

```
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
                <td th:text="${post.createdOn}">post created date</td>
                <td>
                    <a
th:href="@{/admin/posts/{postId}/edit(postId=${post.id})}" class="btn btn-
primary"> Edit</a>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Posts.html

Step 2: Develop Handler Method in PostController

```
@GetMapping("/admin/posts/{postId}/edit")
public String editPostForm(@PathVariable("postId") Long postId,
                           Model model){

    PostDto postDto = postService.findPostById(postId);
    model.addAttribute("post", postDto);
    return "admin/edit_post";
}
```

Step 3: Create Services Method

```
PostDto findPostById(Long postId);
```

PostService.java

```
@Override
public PostDto findPostById(Long postId) {
    Post post = postRepository.findById(postId).get();
    return PostMapper.mapToPostDto(post);
}
```

PostServiceImpl.java

Step 4: Create edit_post.html form

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div th:replace="admin/navbar :: navbar"></div>
<div class="container">
  <br/>
  <br/>
<div class="card" >
  <form method="post"
th:action="@{/admin/posts/{postId}(postId=${post.id})}"
th:object="${post}">
    <input type="hidden" th:field="*{url}">
    <input type="hidden" th:field="*{createdOn}">
    <div class="form-group card-body " >
      <div class="card-header text-center"><h4>Edit Post</h4></div>
      <label >Title</label>
      <input type="text" class="form-control" id="title"
placeholder="Enter Title" name="title" th:field="*{title}">
    </div>
    <p th:if="${#fields.hasErrors('title')}" class="text-danger"
       th:errors="*{title}"></p>
    <div class="form-group card-body">
      <label >Title</label>
      <input type="text" class="form-control" id="shortDescription"
placeholder="Enter Short Description" name="shortDescription"
th:field="*{shortDescription}">
```

```
    </div>
    <p th:if="${#fields.hasErrors('shortDescription')}" class="text-
danger"
       th:errors="*{shortDescription}"></p>
    <div class="form-group card-body">
      <label >Content</label>
      <textarea class="form-control" id="content" rows="10" name="content"
th:field="*{content}"></textarea>
    </div>
    <p th:if="${#fields.hasErrors('content')}" class="text-danger"
       th:errors="*{content}"></p>
    <div class="form-group card-body">
      <button type="submit" class="btn btn-primary card-
body">Submit</button>
    </div>

  </form>

</div>
  <br/>
  <br/>
  <br/>
  <br/>
<div th:replace="admin/footer :: footer"></div>
</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"
        integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
        crossorigin="anonymous"></script>
<script
src="https://cdn.ckeditor.com/ckeditor5/34.2.0/classic/ckeditor.js"></scri
pt>
<script>
                        ClassicEditor
                                .create( document.querySelector(
'#content' ) )
                                .then( editor => {
                                        console.log( editor );
                                } )
                                .catch( error => {
                                        console.error( error );
                                } );
</script>


</body>
</html>
```

edit_posts.html

Step 5: Create handler method in PostController

```
// handler method to handle edit post form submit request
@PostMapping("/admin/posts/{postId}")
public String updatePost(@PathVariable("postId") Long postId,
                         @Valid @ModelAttribute("post") PostDto post,
                         BindingResult result,
                         Model model){
    if(result.hasErrors()){
        model.addAttribute("post", post);
        return "admin/edit_post";
    }

    post.setId(postId);
    postService.updatePost(post);
    return "redirect:/admin/posts";
}
```

PostController.java

Step 6: Create updatePost in PostService

```
void updatePost(PostDto postDto);
```

PostService.java

```
@Override
public void updatePost(PostDto postDto) {
    Post post = PostMapper.mapToPost(postDto);
    postRepository.save(post);
}
```

PostServiceImpl.java

Implementing Delete Feature

Step 1: Add delete Button in posts.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
```

```
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
                <td th:text="${post.createdOn}">post created date</td>
                <td>
                    <a
th:href="@{/admin/posts/{postId}/edit(postId=${post.id})}" class="btn btn-
primary"> Edit</a>
                    <a
th:href="@{/admin/posts/{postId}/delete(postId=${post.id})}" class="btn btn-
danger"> Delete</a>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Posts.html

Step 2: Create Handler Method in PostController

```
// handler method to handle delete post request
@GetMapping("/admin/posts/{postId}/delete")
public String deletePost(@PathVariable("postId") Long postId){
    postService.deletePost(postId);
```

```
    return "redirect:/admin/posts";
}
```

PostController.java

Step 3: Create methods in services layer

```
void deletePost(Long postId);
```

PostService.java

```
@Override
public void deletePost(Long postId) {
    postRepository.deleteById(postId);
}
```

PostServiceImpl.java

Implementing View Feature

Step 1: create view link in posts.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
```

```
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
                <td th:text="${post.createdOn}">post created date</td>
                <td>
                    <a
th:href="@{/admin/posts/{postId}/edit(postId=${post.id})}" class="btn btn-
primary"> Edit</a>
                    <a
th:href="@{/admin/posts/{postId}/delete(postId=${post.id})}" class="btn btn-
danger"> Delete</a>
                    <a
th:href="@{/admin/posts/{postUrl}/view(postUrl=${post.url})}" class="btn btn-
info">View</a>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Posts.html

Step 2: Create Controller Method

```
// handler method to handle view post request
@GetMapping("/admin/posts/{postUrl}/view")
public String viewPost(@PathVariable("postUrl") String postUrl,
                       Model model){
    PostDto postDto = postService.findPostByUrl(postUrl);
    model.addAttribute("post", postDto);
    return "admin/view_post";

}
```

Step 3: Create services layer

```
void deletePost(Long postId);
```

PostService.java

```
@Override
public PostDto findPostByUrl(String postUrl) {
    Post post = postRepository.findByUrl(postUrl).get();
    return PostMapper.mapToPostDto(post);
}
```

PostServiceImpl.java

Step 3: Create view_post.html file

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.
css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>
    <div class="container">
        <div class="row">
            <div class="col-md-8 offset-md-2">
                <h2 th:text="${post.title}"></h2>
                <hr/>
                <h4 th:text="${post.shortDescription}"></h4>
                <hr />
                <div th:utext="${post.content}">
                </div>
            </div>

        </div>
    </div>

    <div th:replace="admin/footer :: footer"></div>
</div>


</body>
</html>
```

View_post.html

Implementing Search Feature

Step 1: Add Search Code in thymeleaf

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <div class="row">
            <div class="col-md-5">
                <div class="form-group">
                    <form class="form-inline"
th:action="@{/admin/posts/search}">
                        <div class="input-group">
                            <input type="text" class="form-control"
name="query" />
                            <span class="input-group-btn">
                            <button class="btn btn-primary" type="submit">
Search</button>
                        </span>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <br />
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
```

```
                <td th:text="${post.createdOn}">post created date</td>
                <td>
                    <a
th:href="@{/admin/posts/{postId}/edit(postId=${post.id})}" class="btn btn-
primary"> Edit</a>
                    <a
th:href="@{/admin/posts/{postId}/delete(postId=${post.id})}" class="btn btn-
danger"> Delete</a>
                    <a
th:href="@{/admin/posts/{postUrl}/view(postUrl=${post.url})}" class="btn btn-
info">View</a>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Step 2: Create handler method in controller layer

```
// handler method to handle search blog posts request
// localhost:8080/admin/posts/search?query=java
@GetMapping("/admin/posts/search")
public String searchPosts(@RequestParam(value = "query") String query,
                          Model model){
    List<PostDto> posts = postService.searchPosts(query);
    model.addAttribute("posts", posts);
    return "admin/posts";
}
```

Step 3: Create JPQL / Native SQL Query in PostRepository

```
package com.blog.repository;

import com.blog.entities.Post;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.List;
import java.util.Optional;

public interface PostRepository  extends JpaRepository<Post,Long> {
    Optional<Post> findByUrl(String url);

    @Query("SELECT p from Post p WHERE " +
```

```
            " p.title LIKE CONCAT('%', :query, '%') OR " +
            " p.shortDescription LIKE CONCAT('%', :query, '%')")
    List<Post> searchPosts(@Param("query") String query);
}
```

PostRepository.java

Step 4: Create following methods in PostService

```
List<PostDto> searchPosts(String query);
```

PostService.java

```
@Override
public List<PostDto> searchPosts(String query) {
    List<Post> posts = postRepository.searchPosts(query);
    return posts.stream()
            .map(PostMapper::mapToPostDto)
            .collect(Collectors.toList());
}
```

PostServiceImpl.java

Implementing Date Format Correction in thymeleaf

Step 1: In posts.html make the following changes

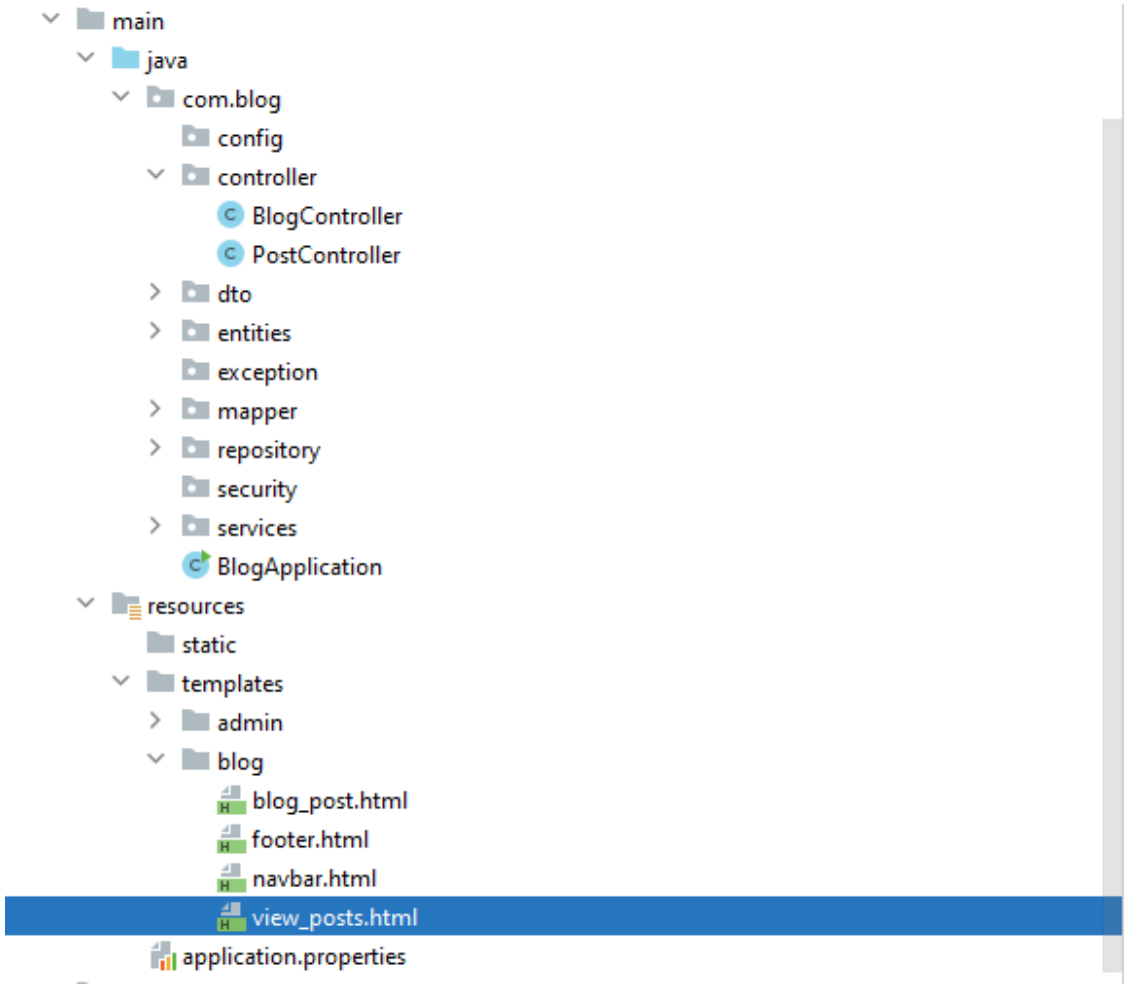
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List Posts</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div th:replace="admin/navbar :: navbar"></div>

    <div class="container">
        <div class="row">
            <h1>List Blog Posts</h1>
        </div>
        <div class="row">
            <div class="col-md-5">
                <div class="form-group">
```

```
                    <form class="form-inline"
th:action="@{/admin/posts/search}">
                        <div class="input-group">
                            <input type="text" class="form-control"
name="query" />
                            <span class="input-group-btn">
                            <button class="btn btn-primary"
type="submit">Search</button>
                        </span>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <br />
        <table class="table table-striped table-hover table-bordered">
            <thead class="table-dark">
            <th style="width:10%">#</th>
            <th style="width:20%" >Post Title</th>
            <th style="width:30%" >Post Short Description</th>
            <th style="width:20%">Post Created On</th>
            <th style="width:20%" >Actions</th>
            </thead>
            <tbody>
            <tr th:each = "post, postStat : ${posts}">
                <td th:text="${postStat.count}">1</td>
                <td th:text="${post.title}">post title</td>
                <td th:text="${post.shortDescription}">post short
description</td>
                <td th:text="${#temporals.format(post.createdOn,'dd MMM
yyyy')}">10 JUL 2022</td>
                <td>
                    <a
th:href="@{/admin/posts/{postId}/edit(postId=${post.id})}" class="btn btn-
primary"> Edit</a>
                    <a
th:href="@{/admin/posts/{postId}/delete(postId=${post.id})}" class="btn btn-
danger"> Delete</a>
                    <a
th:href="@{/admin/posts/{postUrl}/view(postUrl=${post.url})}" class="btn btn-
info">View</a>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
    <br /><br />

    <div th:replace="admin/footer :: footer"></div>
</body>
</html>
```

Developing Client Module



Step 1: Create BlogController class

```
package com.blog.controller;

import com.blog.dto.PostDto;
import com.blog.services.PostService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Controller
public class BlogController {
```

```
    private PostService postService;

    public BlogController(PostService postService) {
        this.postService = postService;
    }

    // handler method to handle http://localhost:8080/
    @GetMapping("/")
    public String viewBlogPosts(Model model){
        List<PostDto> postsResponse = postService.findAllPosts();
        model.addAttribute("postsResponse", postsResponse);
        return "blog/view_posts";
    }
}
```

Step 2: Create view_posts.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div th:replace="blog/navbar :: navbar"></div>
<div class="container">
    <div class="row">
        <div class="col-md-9">
            <div th:each = "post: ${postsResponse}">
                <div class="card">
                    <div class="card-header">
                        <h3>
                            <a
th:href="@{/post/{postUrl}(postUrl=${post.url})}"
                               th:text="${post.title}"></a>
                        </h3>
                        <div>
                            <strong
th:text="${#temporals.format(post.createdOn, 'dd MMMM yyyy')}">

                            </strong>
                        </div>
                    </div>
                    <div class="card-body">
                        <span th:utext="${post.shortDescription}"> </span>
                        <a
th:href="@{/post/{postUrl}(postUrl=${post.url})}">Read more</a>
                    </div>
```

```
                </div>
                <br />
            </div>
</div>

<div th:replace="blog/footer :: footer"></div>
</div>


</body>
</html>
```

Step 3: Update BlogController.java

```
// handler method to handle view post request
@GetMapping("/post/{postUrl}")
private String showPost(@PathVariable("postUrl") String postUrl,
                        Model model){
    PostDto post = postService.findPostByUrl(postUrl);
    model.addAttribute("post", post);
    return "blog/blog_post";
}
```

Step 4: Create blog_post.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css
" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<div th:replace="blog/navbar :: navbar"></div>
<div class="container">
  <div class="row">
    <div class="col-md-9">
      <h2 th:text="${post.title}"></h2>
      <hr/>
      <h4 th:text="${post.shortDescription}"></h4>
      <hr />
      <div th:utext="${post.content}">
      </div>
    </div>

  </div>
```

```
</div>
<div th:replace="blog/footer :: footer"></div>
</body>
</html>
```