**java\_springboot\_restful\_36\_AlbumController – Upgrading the Project**

**Add 2 DTO in album folder**

**PhotoPayloadDTO.java**

package org.studyeasy.SpringRestdemo.payload.album;

import io.swagger.v3.oas.annotations.media.Schema;

import io.swagger.v3.oas.annotations.media.Schema.RequiredMode;

import jakarta.validation.constraints.NotBlank;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@Getter

@Setter

@ToString

@NoArgsConstructor

@AllArgsConstructor

public class PhotoPayloadDTO {

    @NotBlank

    @Schema(description = "Photo name", example = "Selfie", requiredMode = RequiredMode.REQUIRED)

    private String name;

    @NotBlank

    @Schema(description = "Description of the photo", example = "Description",

    requiredMode = RequiredMode.REQUIRED)

    private String description;

}

**PhotoViewDTO.java**

package org.studyeasy.SpringRestdemo.payload.album;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@Setter

@Getter

@ToString

@AllArgsConstructor

@NoArgsConstructor

public class PhotoViewDTO {

    private long id;

    private String name;

    private String desciption;

}

**Updated AlbumController.java**

package org.studyeasy.SpringRestdemo.controller;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Optional;

import javax.imageio.ImageIO;

import org.apache.commons.lang3.RandomStringUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.Resource;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestPart;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

import org.studyeasy.SpringRestdemo.model.Account;

import org.studyeasy.SpringRestdemo.model.Album;

import org.studyeasy.SpringRestdemo.model.Photo;

import org.studyeasy.SpringRestdemo.payload.album.AlbumPayloadDTO;

import org.studyeasy.SpringRestdemo.payload.album.AlbumViewDTO;

import org.studyeasy.SpringRestdemo.payload.album.PhotoDTO;

import org.studyeasy.SpringRestdemo.payload.album.PhotoPayloadDTO;

import org.studyeasy.SpringRestdemo.payload.album.PhotoViewDTO;

import org.studyeasy.SpringRestdemo.service.AccountService;

import org.studyeasy.SpringRestdemo.service.AlbumService;

import org.studyeasy.SpringRestdemo.service.PhotoService;

import org.studyeasy.SpringRestdemo.util.AppUtils.AppUtil;

import org.studyeasy.SpringRestdemo.util.constants.AlbumError;

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.responses.ApiResponse;

import io.swagger.v3.oas.annotations.security.SecurityRequirement;

import io.swagger.v3.oas.annotations.tags.Tag;

import jakarta.validation.Valid;

import lombok.extern.slf4j.Slf4j;

@RestController

@RequestMapping("api/v1")

@Tag(name = "Album Controller", description = "Controller for Album and Photo management")

@Slf4j

public class AlbumController {

    static final String PHOTOS\_FOLDER\_NAME = "photos";

    static final String THUMBNAIL\_FOLDER\_NAME = "thumbnails";

    static final int THUMBNAIL\_WIDTH = 300;

    @Autowired

    private AccountService accountService;

    @Autowired

    private AlbumService albumService;

    @Autowired

    private PhotoService photoService;

    @PostMapping(value = "/albums/add", consumes = "application/json", produces = "application/json")

    @ResponseStatus(HttpStatus.CREATED)

    @ApiResponse(responseCode = "400", description = "Please add valid name a description")

    @ApiResponse(responseCode = "201", description = "Account added")

    @Operation(summary = "Add an Album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<AlbumViewDTO> addAlbum(@Valid @RequestBody AlbumPayloadDTO albumPayloadDTO,

            Authentication authentication) {

        try {

            Album album = new Album();

            album.setName(albumPayloadDTO.getName());

            album.setDescription(albumPayloadDTO.getDescription());

            // Extract Account

            String email = authentication.getName();

            Optional<Account> optionalAccount = accountService.findByEmail(email);

            Account account = optionalAccount.get();

            album.setAccount(account);

            album = albumService.save(album);

            AlbumViewDTO albumViewDTO = new AlbumViewDTO(album.getId(), album.getName(), album.getDescription(), null);

            return ResponseEntity.ok(albumViewDTO);

        } catch (Exception e) {

            log.debug(AlbumError.ADD\_ALBUM\_ERROR.toString() + ": " + e.getMessage());

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

    }

    // List all the albums based on the logged in users

    @GetMapping(value = "/albums", produces = "application/json")

    @ApiResponse(responseCode = "200", description = "List of albums")

    @ApiResponse(responseCode = "401", description = "Token missing")

    @ApiResponse(responseCode = "403", description = "Token Error")

    @Operation(summary = "List album api")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public List<AlbumViewDTO> albums(Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        List<AlbumViewDTO> albums = new ArrayList<>();

        for (Album album : albumService.findByAccount\_id(account.getId())) {

            List<PhotoDTO> photos = new ArrayList<>();

            for(Photo photo : photoService.findByAlbumId(album.getId())){

                String link = "/albums/"+album.getId()+"/photos/"+photo.getId()+"/download-photo";

                photos.add(new PhotoDTO(photo.getId(), photo.getName(), photo.getDescription(), photo.getFileName(),link));

            }

            albums.add(new AlbumViewDTO(album.getId(), album.getName(), album.getDescription(), photos));

        }

        return albums;

    }

    @PostMapping(value = "/albums/{album\_id}/upload\_photos", consumes = { "multipart/form-data" })

    @Operation(summary = "Upload Photo into album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    @ApiResponse(responseCode = "400", description = "Please check the payload of token")

    public ResponseEntity<List<HashMap<String, List<String>>>> photos(

            @RequestPart(required = true) MultipartFile[] files,

            @PathVariable long album\_id, Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        Optional<Album> optionalAlbum = albumService.findById(album\_id);

        Album album;

        if (optionalAlbum.isPresent()) {

            album = optionalAlbum.get();

            if (account.getId() != album.getAccount().getId()) {

                return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

            }

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_GATEWAY).body(null);

        }

        List<String> fileNamesWithSuccess = new ArrayList<>();

        List<String> fileNamesWithError = new ArrayList<>();

        Arrays.asList(files).stream().forEach(file -> {

            // Checking the type of the file is correct or not

            String contentType = file.getContentType();

            if (contentType.equals("image/png")

                    || contentType.equals("image/jpg")

                    || contentType.equals("image/jpeg")) {

                fileNamesWithSuccess.add(file.getOriginalFilename());

                // When we are storing the file in the database, there is a possibility that

                // file name from the user

                // is repeating and if that happens, then file from the user will get repeated

                // and in server will

                // get replaced.

                // In order to prevent that, we need to create a random string.;

                int length = 10;

                boolean useLetters = true;

                boolean useNumbers = true;

                try {

                    String fileName = file.getOriginalFilename();

                    String generatedString = RandomStringUtils.random(length, useLetters, useNumbers);

                    String final\_photo\_name = generatedString + fileName;

                    String absolute\_fileLocation = AppUtil.get\_photo\_upload\_path(final\_photo\_name, PHOTOS\_FOLDER\_NAME,

                            album\_id);

                    Path path = Paths.get(absolute\_fileLocation);

                    Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE\_EXISTING);

                    Photo photo = new Photo();

                    photo.setName(fileName);

                    photo.setFileName(final\_photo\_name);

                    photo.setOriginalFileName(fileName);

                    photo.setAlbum(album);

                    photoService.save(photo);

                    BufferedImage thumbImg = AppUtil.getThumbnail(file, THUMBNAIL\_WIDTH);

                    File thumbnail\_location = new File(

                            AppUtil.get\_photo\_upload\_path(final\_photo\_name, THUMBNAIL\_FOLDER\_NAME, album\_id));

                    ImageIO.write(thumbImg, file.getContentType().split("/")[1], thumbnail\_location);

                } catch (Exception e) {

                    log.debug(AlbumError.PHOTO\_UPLOAD\_ERROR.toString() + ": " + e.getMessage());

                    fileNamesWithError.add(file.getOriginalFilename());

                }

            } else {

                fileNamesWithError.add(file.getOriginalFilename());

            }

        });

        HashMap<String, List<String>> result = new HashMap<>();

        result.put("SUCCESS", fileNamesWithSuccess);

        result.put("ERRORS", fileNamesWithError);

        List<HashMap<String, List<String>>> response = new ArrayList<>();

        response.add(result);

        return ResponseEntity.ok(response);

    }

    @GetMapping("albums/{album\_id}/photos/{photo\_id}/download-photo")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<?> downloadPhoto(@PathVariable("album\_id") long album\_id,

            @PathVariable("photo\_id") long photo\_id, Authentication authentication) {

        return downloadFile(album\_id, photo\_id, PHOTOS\_FOLDER\_NAME, authentication);

    }

    @GetMapping("albums/{album\_id}/photos/{photo\_id}/download-thumbnail")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<?> downloadThumbnail(@PathVariable("album\_id") long album\_id,

            @PathVariable("photo\_id") long photo\_id, Authentication authentication) {

        return downloadFile(album\_id, photo\_id, PHOTOS\_FOLDER\_NAME, authentication);

    }

    public ResponseEntity<?> downloadFile(long album\_id, long photo\_id, String folder\_name,

            Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        Optional<Album> optionaAlbum = albumService.findById(album\_id);

        Album album;

        if (optionaAlbum.isPresent()) {

            album = optionaAlbum.get();

            if (account.getId() != album.getAccount().getId()) {

                return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

            }

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

        Optional<Photo> optionalPhoto = photoService.findById(photo\_id);

        if (optionalPhoto.isPresent()) {

            Photo photo = optionalPhoto.get();

            if (photo.getAlbum().getId() != album\_id) {

                return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

            }

            Resource resource = null;

            try {

                resource = AppUtil.getFileAsResource(album\_id, folder\_name, photo.getFileName());

            } catch (IOException e) {

                return ResponseEntity.internalServerError().build();

            }

            if (resource == null) {

                return new ResponseEntity<>("File not found", HttpStatus.NOT\_FOUND);

            }

            String contentType = "application/octet-stream";

            String headerValue = "attachment; filename=\"" + photo.getOriginalFileName() + "\"";

            return ResponseEntity.ok()

                    .contentType(MediaType.parseMediaType(contentType))

                    .header(HttpHeaders.CONTENT\_DISPOSITION, headerValue)

                    .body(resource);

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

    }

    //Album By Id

    @GetMapping(value = "/albums/{album\_id}", produces = "application/json")

    @ApiResponse(responseCode = "200", description = "List of albums")

    @ApiResponse(responseCode = "401", description = "Token missing")

    @ApiResponse(responseCode = "403", description = "Token Error")

    @Operation(summary = "List album by album ID")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<AlbumViewDTO> albums\_by\_id(@PathVariable long album\_id, Authentication authentication) {

        String email = authentication.getName();

        Optional<Account> optionalAccount = accountService.findByEmail(email);

        Account account = optionalAccount.get();

        Optional<Album> optionalAlbum = albumService.findById(album\_id);

        Album album;

        if (optionalAlbum.isPresent()) {

            album = optionalAlbum.get();

        } else {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

        if (account.getId() != album.getAccount().getId()) {

            return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

        }

        List<PhotoDTO> photos = new ArrayList<>();

        for (Photo photo : photoService.findByAlbumId(album.getId())) {

            String link = "/albums/"+album.getId()+"/photos/"+photo.getId()+"/download-photo";

            photos.add(new PhotoDTO(photo.getId(), photo.getName(), photo.getDescription(), photo.getFileName(), link));

        }

        AlbumViewDTO albumViewDTO = new AlbumViewDTO(album.getId(), album.getName(), album.getDescription(), photos);

        return ResponseEntity.ok(albumViewDTO);

    }

    //Update Album

    @PutMapping(value = "/albums/{album\_id}/update", consumes = "application/json", produces = "application/json")

    @ResponseStatus(HttpStatus.CREATED)

    @ApiResponse(responseCode = "400", description = "Please add valid name a description")

    @ApiResponse(responseCode = "204", description = "Album update")

    @Operation(summary = "Update an Album")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<AlbumViewDTO> update\_Album(@Valid @RequestBody AlbumPayloadDTO albumPayloadDTO,

            @PathVariable long album\_id, Authentication authentication) {

        try {

            String email = authentication.getName();

            Optional<Account> optionalAccount = accountService.findByEmail(email);

            Account account = optionalAccount.get();

            Optional<Album> optionaAlbum = albumService.findById(album\_id);

            Album album;

            if (optionaAlbum.isPresent()) {

                album = optionaAlbum.get();

                if (account.getId() != album.getAccount().getId()) {

                    return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

                }

            } else {

                return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

            }

            album.setName(albumPayloadDTO.getName());

            album.setDescription(albumPayloadDTO.getDescription());

            album = albumService.save(album);

            List<PhotoDTO> photos = new ArrayList<>();

            for(Photo photo: photoService.findByAlbumId(album.getId())){

                String link = "/albums/"+album.getId()+"/photos/"+photo.getId()+"/download-photo";

                photos.add(new PhotoDTO(photo.getId(), photo.getName(), photo.getDescription(),

                photo.getFileName(), link));

            }

            AlbumViewDTO albumViewDTO = new AlbumViewDTO(album.getId(), album.getName(), album.getDescription(), photos);

            return ResponseEntity.ok(albumViewDTO);

        } catch (Exception e) {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

    }

    @PutMapping(value = "albums/{album\_id}/photos/{photo\_id}/update", consumes = "application/json", produces = "application/json")

    @ResponseStatus(HttpStatus.CREATED)

    @ApiResponse(responseCode = "400", description = "Please add valid name a description")

    @ApiResponse(responseCode = "204", description = "Album update")

    @Operation(summary = "Update a photo")

    @SecurityRequirement(name = "studyeasy-demo-api")

    public ResponseEntity<PhotoViewDTO> update\_photo(@Valid @RequestBody PhotoPayloadDTO photoPayloadDTO,

            @PathVariable long album\_id, @PathVariable long photo\_id,Authentication authentication) {

        try {

            String email = authentication.getName();

            Optional<Account> optionalAccount = accountService.findByEmail(email);

            Account account = optionalAccount.get();

            Optional<Album> optionaAlbum = albumService.findById(album\_id);

            Album album;

            if (optionaAlbum.isPresent()) {

                album = optionaAlbum.get();

                if (account.getId() != album.getAccount().getId()) {

                    return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

                }

            } else {

                return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

            }

            Optional<Photo> optionalPhoto = photoService.findById(photo\_id);

            if(optionalPhoto.isPresent()){

                Photo photo = optionalPhoto.get();

                if (photo.getAlbum().getId() != album\_id) {

                    return ResponseEntity.status(HttpStatus.FORBIDDEN).body(null);

                }

                photo.setName(photoPayloadDTO.getName());

                photo.setDescription(photoPayloadDTO.getDescription());

                photoService.save(photo);

                PhotoViewDTO photoViewDTO = new PhotoViewDTO(photo.getId(), photoPayloadDTO.getName(), photoPayloadDTO.getDescription());

                return ResponseEntity.ok(photoViewDTO);

            }else{

                return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

            }

        } catch (Exception e) {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

    }

}