**java\_springboot\_restful\_30\_AlbumController – Build Photo Upload for Album With Thumbnails**

**Updated AppUtil.java**

package org.studyeasy.SpringRestdemo.util.AppUtils;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

//Receiving the filename and also the album\_id

public class AppUtil {

    public static String get\_photo\_upload\_path(String fileName, String folder\_name ,long album\_id) throws IOException {

        String path = "src\\main\\resources\\static\\uploads\\" + album\_id + "\\" +folder\_name;

        Files.createDirectories(Paths.get(path));

        return new File(path).getAbsolutePath() + "\\" + fileName;

    }

}

**Updated AlbumError.java**

package org.studyeasy.SpringRestdemo.util.constants;

public enum AlbumError {

    ADD\_ALBUM\_ERROR,

    PHOTO\_UPLOAD\_ERROR

}

**Add this in pom.xml**

 <!-- Adding Scalar-->

 <dependency>

     <groupId>org.imgscalr</groupId>

     <artifactId>imgscalr-lib</artifactId>

      <version>4.2</version>

      <type>jar</type>

      <scope>compile</scope>

</dependency>

**Updated AppUtil.java**

package org.studyeasy.SpringRestdemo.util.AppUtils;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import javax.imageio.ImageIO;

import org.imgscalr.Scalr;

import org.springframework.web.multipart.MultipartFile;

import java.awt.image.BufferedImage;

//Receiving the filename and also the album\_id

public class AppUtil {

    public static String get\_photo\_upload\_path(String fileName, String folder\_name, long album\_id) throws IOException {

        String path = "src\\main\\resources\\static\\uploads\\" + album\_id + "\\" + folder\_name;

        Files.createDirectories(Paths.get(path));

        return new File(path).getAbsolutePath() + "\\" + fileName;

    }

    public static BufferedImage getThumbnail(MultipartFile orginalFile, Integer width) throws IOException {

        BufferedImage thumbImg = null;

        BufferedImage img = ImageIO.read(orginalFile.getInputStream());

        thumbImg = Scalr.resize(img, Scalr.Method.AUTOMATIC, Scalr.Mode.AUTOMATIC, width, Scalr.OP\_ANTIALIAS);

        return thumbImg;

    }

}

**Updated photo method in AlbumController.java**

  @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);

    }