

## java springboot restful 32 AlbumController – Download Thumbnail API

### 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.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
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.auth.album.AlbumPayloadDTO;
import org.studyeasy.SpringRestdemo.payload.auth.album.AlbumViewDTO;
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());
        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())) {
        albums.add(new AlbumViewDTO(album.getId(), album.getName(),
album.getDescription()));
    }
    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();
            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);
    }
}
}

```

## Output

### Token generated



## Jwt.io says it is user and authorize through this

The screenshot shows the jwt.io website interface. At the top, there's a navigation bar with links for Debugger, Libraries, Introduction, and Ask. Below the navigation bar, there's a section for "Encoded" and "Decoded" tokens. The "Encoded" section contains a long string of characters representing a JWT token. The "Decoded" section shows the token's structure, including the header, payload, and signature verification details.

**Encoded**

```
eyJraWQ1OjI3ZTY3YjI0MC0zMDNkLTRkYmQtYjM5MS0zNjM4MzNmZjQwYzI1LCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoiaXNlckB1c2VyLnNvbSI9ImV4cCI6MTcyNDANdIyN1wWF0IjozNzI0MDANjI2LCJzY29wZSI6I1VTRVIiFQ.ZoLLX-ysVJmBbcx2WrW1cpXQSDOMFu4K4c5rNZFn61HiItQpVuphLbufBnwhkxG3ptcW9E_ozQLqPfZkYTulTdj6VjNhgz0kBB0P6KELg_bvVR_vBnoY6JDaW7Je21eN01dxytiSkhcJJXY4etdVSmvHTTj6gDKzWX77B0dqoPonfw-LcYQ5AYpUkPuIBoPHI51NfkkG0GxG8x8U43Sc3trKRFPyd1v_ewFyVLoLVE3FpPueXQU2ew21Pw0gUOdY13ICgK4nkhVVFAmhJQUy6B-a7vWm5wScMYJckJcm3HsfINJme1o3zVxRC1qAh2T3wWL5kXjOjTY26PmVBw
```

**Decoded**

HEADER: ALGORITHM & TOKEN TYPE

```
{  "kid": "7e67b248-383d-4dbd-b391-363833ff48cc",  "alg": "RS256"}
```

PAYLOAD: DATA

```
{  "iss": "self",  "sub": "user@user.com",  "exp": 1724004226,  "iat": 1724000626,  "scope": "USER"}
```

VERIFY SIGNATURE

```
RSASHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),
```

## Add Album

The screenshot shows the Swagger UI interface for the 'Add Album' endpoint. The interface includes a 'Execute' button and a 'Clear' button. Below these buttons, the 'Responses' section displays the request and response details. The request is a POST to the endpoint 'http://localhost:8080/api/v1/albums/add' with a JSON body containing album details. The response is a 200 status code with a JSON body containing the created album details.

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'POST' \  "http://localhost:8080/api/v1/albums/add" \  -H 'accept: application/json' \  -H 'Authorization: Bearer eyJraWQ1OjI3ZTY3YjI0MC0zMDNkLTRkYmQtYjM5MS0zNjM4MzNmZjQwYzI1LCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwic3ViIjoiaXNlckB1c2VyLnNvbSI9ImV4cCI6MTcyNDANdIyN1wWF0IjozNzI0MDANjI2LCJzY29wZSI6I1VTRVIiFQ.ZoLLX-ysVJmBbcx2WrW1cpXQSDOMFu4K4c5rNZFn61HiItQpVuphLbufBnwhkxG3ptcW9E_ozQLqPfZkYTulTdj6VjNhgz0kBB0P6KELg_bvVR_vBnoY6JDaW7Je21eN01dxytiSkhcJJXY4etdVSmvHTTj6gDKzWX77B0dqoPonfw-LcYQ5AYpUkPuIBoPHI51NfkkG0GxG8x8U43Sc3trKRFPyd1v_ewFyVLoLVE3FpPueXQU2ew21Pw0gUOdY13ICgK4nkhVVFAmhJQUy6B-a7vWm5wScMYJckJcm3HsfINJme1o3zVxRC1qAh2T3wWL5kXjOjTY26PmVBw' \  -H 'Content-Type: application/json' \  -d '{  "name": "Travel",  "description": "description"  }'
```

**Request URL**

```
http://localhost:8080/api/v1/albums/add
```

**Server response**

**Code** **Details**

**200** **Response body**

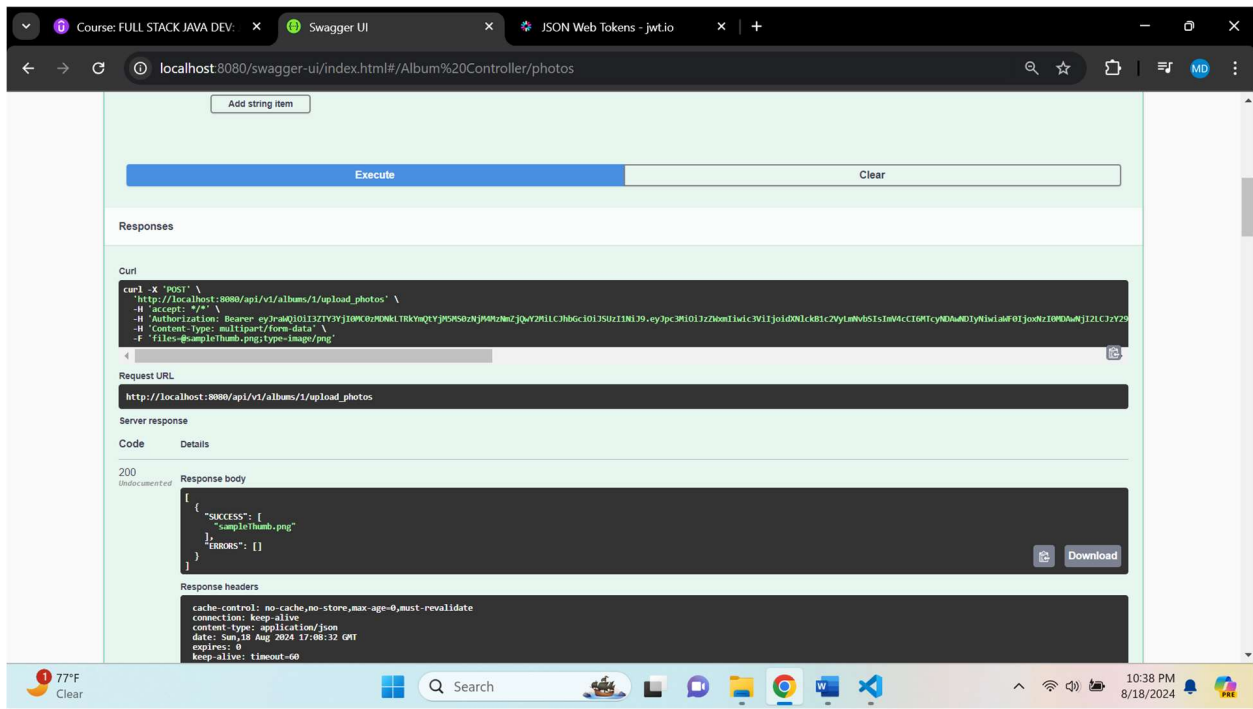
```
{  "id": 1,  "name": "Travel",  "description": "description"}
```

**Response headers**

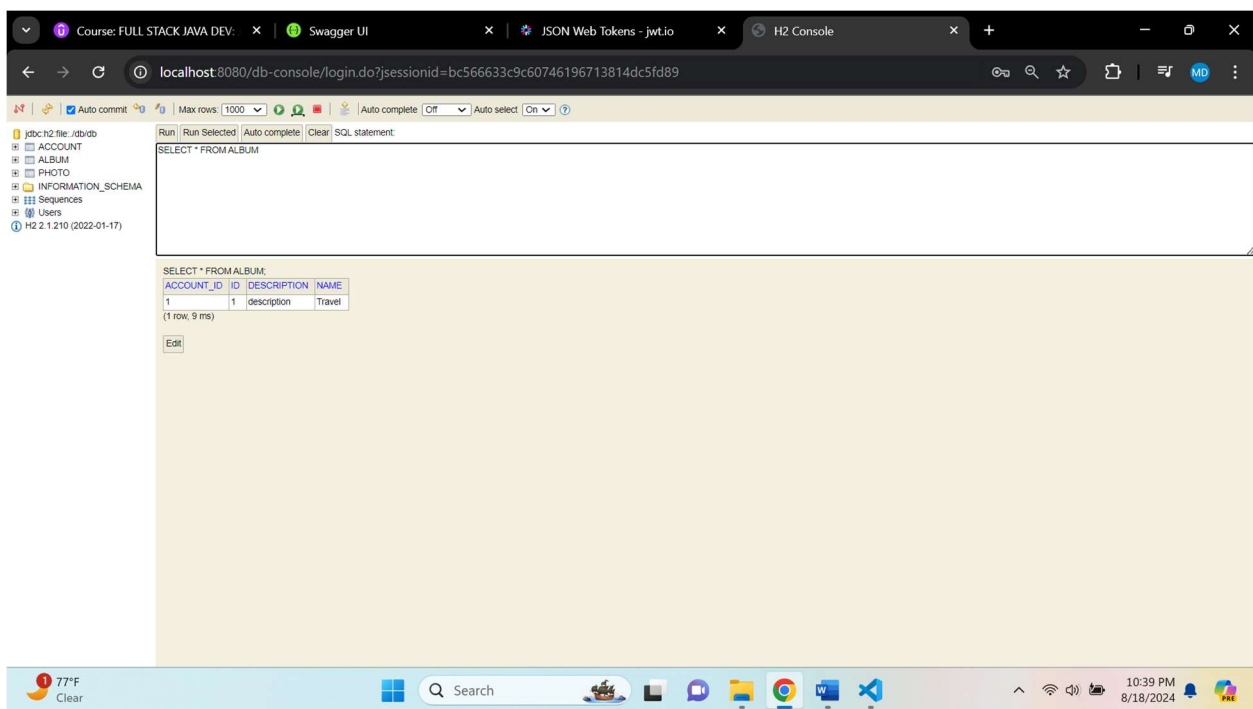
```
cache-control: no-cache, no-store, max-age=0, must-revalidate  connection: keep-alive  content-type: application/json  date: Sun, 18 Aug 2024 17:06:06 GMT  expires: 0  keep-alive: timeout=60  pragma: no-cache
```



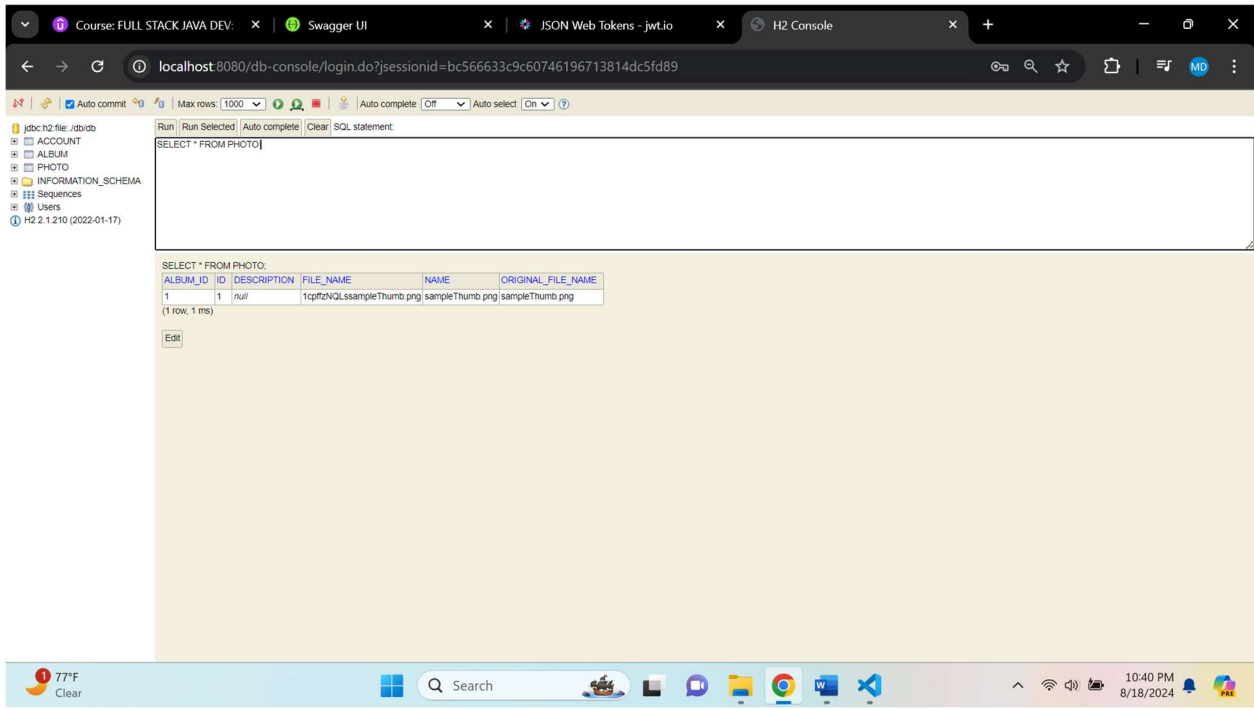
Id is 1 and upload photo – It will show added successfully and then add one image/ upload one image to the album where id is 1



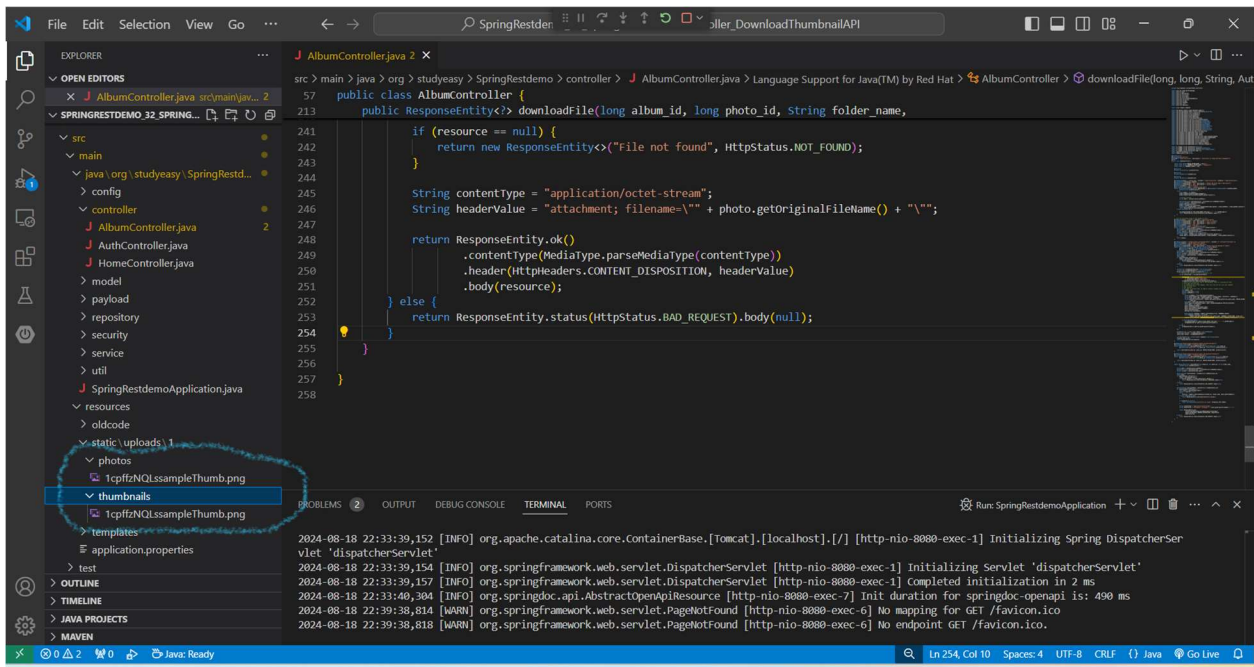
## Db Console – Album



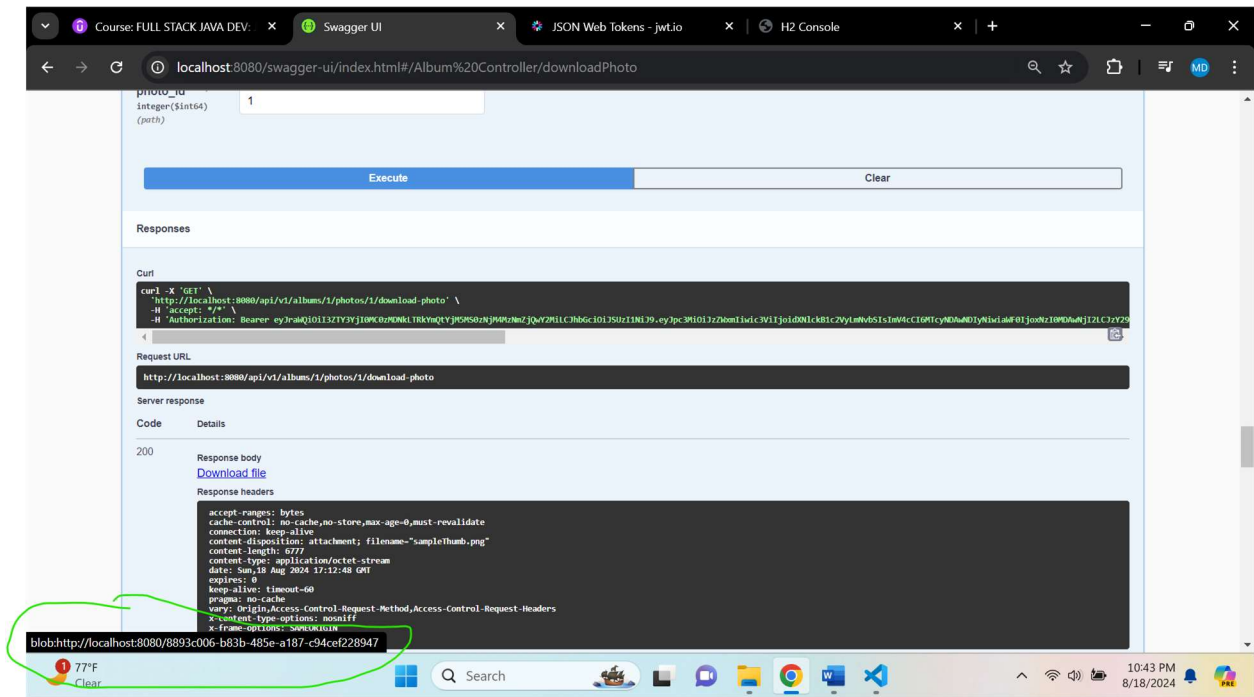
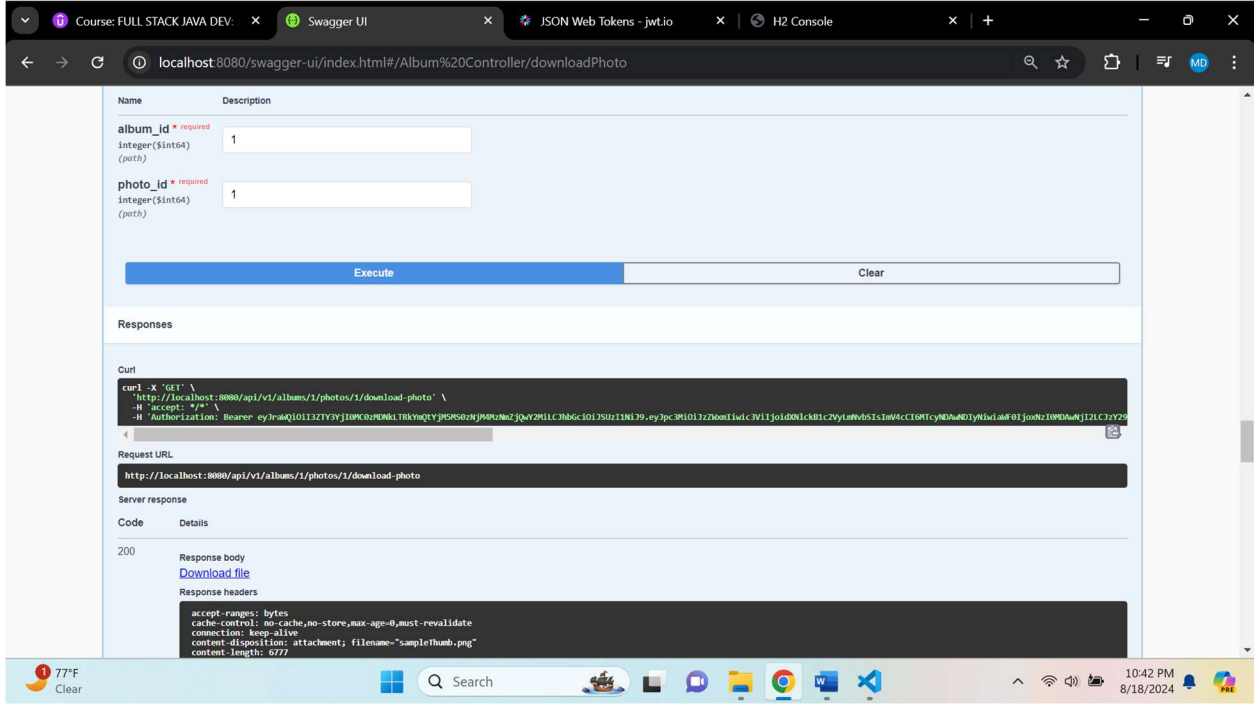
## Db Console – Photo



## In the Code



**Download Photo**



The screenshot displays the Swagger UI interface for a REST API. The 'Parameters' section is active, showing two required path parameters: 'album\_id' and 'photo\_id', both with integer values set to '1'. Below the parameters is an 'Execute' button. The 'Responses' section shows the raw HTTP response, including headers like 'Accept' and 'Authorization', and the request URL 'http://localhost:8080/api/v1/albums/1/photos/1/download-thumbnail'. The server response is a 200 status code with a 'Response body' link and a 'Download file' link.

[illegible]