Setting up Cloudinary in Spring Boot

Step 1: Add Cloudinary Dependency

Add the Cloudinary dependency to your pom.xml file:

xml

```
<dependency>
    <groupId>com.cloudinary</groupId>
    <artifactId>cloudinary-http44</artifactId>
    <version>1.31.0</version>
</dependency>
```

Step 2: Configure Cloudinary

Create a configuration class CloudinaryConfiguration to set up Cloudinary:

java

```
import com.cloudinary.Cloudinary;
import com.cloudinary.utils.ObjectUtils;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class CloudinaryConfiguration {

    @Value("${cloudinary.cloud_name}")
    private String cloudName;

    @Value("${cloudinary.api_key}")
    private String apiKey;

    @Value("${cloudinary.api_secret}")
    private String apiSecret;

    @Bean
    public Cloudinary cloudinary() {
        return new Cloudinary(ObjectUtils.asMap(
                "cloud_name", cloudName,
                "api_key", apiKey,
                "api_secret", apiSecret));
    }
}
```

Step 3: Create Image Entity and Repository

Define the Image entity and its repository:

java

```
import jakarta.persistence.*;
import lombok.Data;

import java.util.UUID;

@Entity
@Data
@Table(name = "images")
public class Image {
    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    @Column(name = "id")
    private UUID id;

    @Column(name = "name_image")
    private String name;

    @Column(name = "url_image")
    private String url;
}

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface ImageRepository extends JpaRepository<Image, UUID> {
}
```

Step 4: Implement Cloudinary Service

Create the CloudinaryService interface and its implementation:

java

```
import org.springframework.web.multipart.MultipartFile;

import java.util.Map;

public interface CloudinaryService {
    String uploadFile(MultipartFile file, String folderName);
}

import com.cloudinary.Cloudinary;
import org.springframework.stereotype.Service;
```

```
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

@Service
public class CloudinaryServiceImpl implements CloudinaryService {

    @Resource
    private Cloudinary cloudinary;

    @Override
    public String uploadFile(MultipartFile file, String folderName) {
        try {
            HashMap<Object, Object> options = new HashMap<>();
            options.put("folder", folderName);
            Map uploadedFile = cloudinary.uploader().upload(file.getBytes(), options);
            String publicId = (String) uploadedFile.get("public_id");
            return cloudinary.url().secure(true).generate(publicId);

        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Step 5: Implement Image Service

Create the ImageService interface and its implementation:

java

```
import com.cloudinary.springboot.dto.ImageModel;
import org.springframework.http.ResponseEntity;

import java.util.Map;

public interface ImageService {
    ResponseEntity<Map> uploadImage(ImageModel imageModel);
}

import com.cloudinary.springboot.dto.ImageModel;
import com.cloudinary.springboot.entity.Image;
import com.cloudinary.springboot.repository.ImageRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
```

```
import java.util.Map;

@Service
public class ImageServiceImpl implements ImageService {

    @Autowired
    private CloudinaryService cloudinaryService;

    @Autowired
    private ImageRepository imageRepository;

    @Override
    public ResponseEntity<Map> uploadImage(ImageModel imageModel) {
        try {
            if (imageModel.getName().isEmpty() || imageModel.getFile().isEmpty()) {
                return ResponseEntity.badRequest().build();
            }

            Image image = new Image();
            image.setName(imageModel.getName());
            image.setUrl(cloudinaryService.uploadFile(imageModel.getFile(), "folder_1"));
            if (image.getUrl() == null) {
                return ResponseEntity.badRequest().build();
            }
            imageRepository.save(image);
            return ResponseEntity.ok().body(Map.of("url", image.getUrl()));
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Step 6: Create DTO and Controller

Define the DTO class and the controller:

java

```
import lombok.Data;
import org.springframework.web.multipart.MultipartFile;

@Data
public class ImageModel {
    private String name;
    private MultipartFile file;
}
```

```
import com.cloudinary.springboot.dto.ImageModel;
import com.cloudinary.springboot.service.ImageService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController
public class ImageController {

    @Autowired
    private ImageService imageService;

    @PostMapping("/upload")
    public ResponseEntity<Map> upload(ImageModel imageModel) {
        try {
            return imageService.uploadImage(imageModel);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Step 7: Usage

Now you can use the /upload endpoint to upload images. Make a POST request to this endpoint with the name and file parameters in the request body.

This setup allows you to easily integrate Cloudinary into your Spring Boot project for image uploading and retrieval. Customize the configurations and services as per your requirements