Spring Boot REST Upload

Einführung

Der Upload von Dateien ist eine Grundanforderung von Web Anwendungen und damit auch von REST Services. Dieser Blog zeigt auf, wie man MultipartFile Daten mit REST Services programmiert und testet. Wir arbeiten mit der Eclipse IDE und den Spring Tools 4.

Spring Boot Application

Zuerst öffnen wir die Eclipse IDE und erstellen mit dem Spring Tools 4 Plugin eine Spring Boot Application von Grund auf: Mit dem Abschluss des Projekts erhalten Sie eine lauffähige leere Spring Boot Application.

Upload Controller

Um Dateien als Upload einem Spring REST Service zur Verfügung zu stellen, verwenden wir eine Instanz der Spring Klasse org.springframework.web.multipart.MultipartFile. Es handelt sich hier um eine InputStreamSource Klasse, welche ein File als Multipart Request repräsentiert. Das folgende Listing zeigt den UploadController mit der Methode uploadDokument und einem MultipartFile Parameter:package ch.std.rest.upload.controller;

import java.nio.file.Path;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestPart;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import ch.std.rest.upload.dto.RequestUploadDTO;
import ch.std.rest.upload.dto.ResponseUploadDTO;
import ch.std.rest.upload.service.FileService;

@RestController
@RequestMapping(path="/rest")
public class UploadController {

 Logger logger = LoggerFactory.getLogger(UploadController.class);

 private FileService fileService;

 public UploadController(FileService fileService) {
  this.fileService = fileService;
 }

 @PostMapping(path = "upload", produces = { MediaType.APPLICATION_JSON_VALUE })
 public ResponseUploadDTO uploadDokument(@RequestPart("file") MultipartFile file) {
  String fileName = file.getOriginalFilename();
  ResponseUploadDTO responseUploadDTO = new ResponseUploadDTO();
  try {
  Path out = this.fileService.save(fileName, file.getInputStream());
  responseUploadDTO.setSuccess("file uploaded");
 responseUploadDTO.setPath(out);
  } catch (Exception e) {
 responseUploadDTO.setFailed(e.getMessage());
  }
  return responseUploadDTO;
 }

}
Das Listing verwendet als Response die Klasse ResponseUploadDTO:package ch.std.rest.upload.dto;

import java.nio.file.Path;

public class ResponseUploadDTO {

 private int status;
 private String message;
 private Path path;

 public void setSuccess(String message) {
  this.status = 0;
 this.message = message;
 }

 public void setFailed(String message) {
 this.status = 1;
  this.message = message;
 }

 public String getMessage() {
  return message;
 }

 public Path getPath() {
  return path;
 }

 public void setPath(Path path) {
  this.path = path;
 }

 public int getStatus() {
  return status;
 }

 @Override
 public String toString() {
  return "ResponseUploadDTO [status=" + status + ", message=" + message + ", path=" + path + "]";
 }

}Die Verarbeitung des Uploads erfolgt über den FileService:package ch.std.rest.upload.service;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.file.Files;
import java.nio.file.Path;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

@Service
public class FileService {

 Logger logger = LoggerFactory.getLogger(FileService.class);

 @Value("${upload.service.path}")
 private String rootPath;

 public

FileService() {
 }

 public Path save(String fileName, InputStream is) throws IOException {
  byte[] buffer = new byte[1 << 20];
  int bytesRead = -1;
  int bytesWritten = 0;
  Path out = Path.of(rootPath, fileName);
  try (OutputStream os = Files.newOutputStream(out)) {
  while((bytesRead = is.read(buffer)) > 0) {
  os.write(buffer, 0, bytesRead);
  bytesWritten += bytesRead;
 }
  }
  logger.info("file " + fileName + " written to path " + out + ", bytes written " + bytesWritten);
  return out;
 }

}Der File Service speichert die uploaded Datei relativ zum Pfad definiert durch das Property "upload.service.path". Solches definieren wir in der application.properteies Datei:upload.service.path=/tmp

Upload Controller Integration Test

Das folgende Listing zeigt den Integration Test für den UploadController upload REST Endpoint:package ch.std.demo.upload.rest.test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.util.LinkedMultiValueMap;

import ch.std.demo.upload.dto.RequestUploadDTO;
import ch.std.demo.upload.dto.ResponseUploadDTO;

@ExtendWith(SpringExtension.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
public class UploadServiceIntegrationTests {

 @Autowired
 private TestRestTemplate restTemplate;

 @Test
 public void contextLoads() {
 }

 @Test
 public void testUploadController() throws Exception {
  var multipart = new LinkedMultiValueMap<>();
  multipart.add("file", new org.springframework.core.io.ClassPathResource("dummy.pdf"));

  final ResponseEntity<ResponseUploadDTO> post = this.restTemplate.postForEntity("/rest/upload",
  new HttpEntity<>(multipart, headers()), ResponseUploadDTO.class);

 assertEquals(HttpStatus.OK, post.getStatusCode());
  System.out.println("response = " + post.getBody());
 }

 private HttpHeaders headers() {
  HttpHeaders headers = new HttpHeaders();
 headers.setContentType(MediaType.MULTIPART_FORM_DATA);
  return headers;
 }
}Der Integration Test verwendet die Datei dummy.pdf, welche im Pfad "src/test/resources" abgelegt ist. Die Datei dummy.pdf können Sie via Hyperlink downloaden und in das Projekt integrieren. Der Integration Unit Test sollte korrekt funktionieren.

Upload Controller mit DTO

Das folgende Listing zeigt einen REST Endpoint mit zusätzlichem Request DTO Objekt:@PostMapping(path = "uploaddto", produces = { MediaType.APPLICATION_JSON_VALUE })
public ResponseUploadDTO uploadDokumentWithDTO(@RequestPart("dto") RequestUploadDTO dto, @RequestPart("file") MultipartFile file) {
  String fileName = dto.getFileName();
 ResponseUploadDTO responseUploadDTO = new ResponseUploadDTO();
  try {
  Path out = this.fileService.save(fileName, file.getInputStream());
 responseUploadDTO.setSuccess("file uploaded");
 responseUploadDTO.setPath(out);
  } catch (Exception e) {
 responseUploadDTO.setFailed(e.getMessage());
  }
  return responseUploadDTO;
}Den zugehörigen Integration Test sehen Sie im folgenden Listing:@Test
public void testUploadDTOController() throws Exception {
  var multipart = new LinkedMultiValueMap<>();
  RequestUploadDTO requestUploadDTO = new RequestUploadDTO();
  requestUploadDTO.setFileName("dummydto.pdf");
 multipart.add("dto", requestUploadDTO);
  multipart.add("file", new

org.springframework.core.io.ClassPathResource("dummy.pdf"));

  final ResponseEntity<ResponseUploadDTO> post = this.restTemplate.postForEntity("/rest/uploaddto",
    new HttpEntity<>(multipart, headers()), ResponseUploadDTO.class);

 assertEquals(HttpStatus.OK, post.getStatusCode());
  System.out.println("response = " + post.getBody());
}Die RequestDTO Klasse finden Sie hier:package ch.std.rest.upload.dto;

public class RequestUploadDTO {

 private String fileName;

 public RequestUploadDTO() {
 }

 public String getFileName() {
  return fileName;
 }

 public void setFileName(String fileName) {
  this.fileName = fileName;
 }
}Der Test sollte korrekt funktionieren.

Full Sample Download

Das gesamte Beispiel finden Sie unter dem Link springbootrestupload.zip.

Feedback

War dieser Blog für Sie wertvoll. Wir danken für jede Anregung und Feedback

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/dummy.pdf