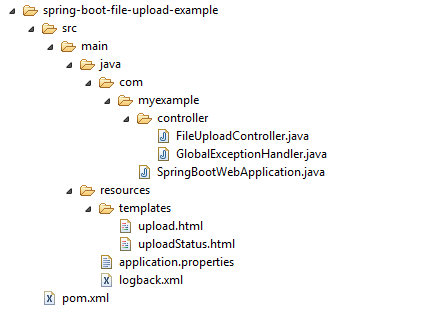
Introduction:

This document describes about how to implement a Restful API in spring-boot application. API to upload a file with a few meta-data fields. Persist meta-data in persistent store file system and store the content on a file system

Prerequisites:

1. Spring Boot 1.4.3.RELEASE
2. Spring 4.3.5.RELEASE
3. Thymeleaf
4. Maven
5. Embedded Tomcat 8.5.6

Project Structure:



## Project Dependency

Only Spring boot dependencies, no extra libraries for file uploading are necessary

Pom.xml:

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*  *http://maven.apache.org/maven-v4\_0\_0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.myexample</groupId>  <artifactId>spring-boot-file-upload</artifactId>  <packaging>jar</packaging>  <version>1.0</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.3.RELEASE</version>  </parent>  <properties>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency>  <!-- hot swapping, disable cache for template, enable live reload -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <optional>true</optional>  </dependency>  </dependencies>  <build>  <plugins>  <!-- Package as an executable jar/war -->  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

## File Upload Example

As we are using Spring boot file upload, zero configuration is required .

FileUploadController.java: This controller is used for file upload processing

|  |
| --- |
| package com.myexample.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.multipart.MultipartFile;  import org.springframework.web.servlet.mvc.support.RedirectAttributes;  import java.io.IOException;  import java.nio.file.Files;  import java.nio.file.Path;  import java.nio.file.Paths;  @Controller  public class FileUploadController {  //Save the uploaded file to this folder  private static String UPLOADED\_FOLDER = "F://temp//";  @GetMapping("/")  public String index() {  return "upload";  }  @PostMapping("/upload") // //new annotation since 4.3  public String singleFileUpload(@RequestParam("file") MultipartFile file,  RedirectAttributes redirectAttributes) {  if (file.isEmpty()) {  redirectAttributes.addFlashAttribute("message", "Please select a file to upload");  return "redirect:uploadStatus";  }  try {  // Get the file and save it somewhere  byte[] bytes = file.getBytes();  Path path = Paths.get(UPLOADED\_FOLDER + file.getOriginalFilename());  Files.write(path, bytes);  redirectAttributes.addFlashAttribute("message",  "You successfully uploaded '" + file.getOriginalFilename() + "'");  } catch (IOException e) {  e.printStackTrace();  }  return "redirect:/uploadStatus";  }  @GetMapping("/uploadStatus")  public String uploadStatus() {  return "uploadStatus";  }  } |

GlobalExceptionHandler.java-this is to handle all the exceptions raised by application in a grace full way.

|  |
| --- |
| package com.myexample.controller;  import org.springframework.web.bind.annotation.ControllerAdvice;  import org.springframework.web.bind.annotation.ExceptionHandler;  import org.springframework.web.multipart.MultipartException;  import org.springframework.web.servlet.mvc.support.RedirectAttributes;  @ControllerAdvice  public class GlobalExceptionHandler {    @ExceptionHandler(MultipartException.class)  public String handleError1(MultipartException e, RedirectAttributes redirectAttributes) {  redirectAttributes.addFlashAttribute("message", e.getCause().getMessage());  return "redirect:/uploadStatus";  }  } |

SpringBootWebApplication.java:A start up class to initiate spring boot application

Note: If you deployed to Tomcat, configure the maxSwallowSize to avoid this [Tomcat connection reset issue](http://www.mkyong.com/spring/spring-file-upload-and-connection-reset-issue/). For embedded Tomcat, declares a TomcatEmbeddedServletContainerFactory like the following :

|  |
| --- |
| package com.myexample;  import org.apache.coyote.http11.AbstractHttp11Protocol;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.boot.context.embedded.tomcat.TomcatConnectorCustomizer;  import org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainerFactory;  import org.springframework.context.annotation.Bean;  @SpringBootApplication  public class SpringBootWebApplication {  private int maxUploadSizeInMb = 10 \* 1024 \* 1024; // 10 MB  public static void main(String[] args) throws Exception {  SpringApplication.run(SpringBootWebApplication.class, args);  }  //Tomcat large file upload connection reset  @Bean  public TomcatEmbeddedServletContainerFactory tomcatEmbedded() {  TomcatEmbeddedServletContainerFactory tomcat = new TomcatEmbeddedServletContainerFactory();  tomcat.addConnectorCustomizers((TomcatConnectorCustomizer) connector -> {  if ((connector.getProtocolHandler() instanceof AbstractHttp11Protocol<?>)) {  //-1 means unlimited  ((AbstractHttp11Protocol<?>) connector.getProtocolHandler()).setMaxSwallowSize(-1);  }  });  return tomcat;  }  } |

## Multipart File Size

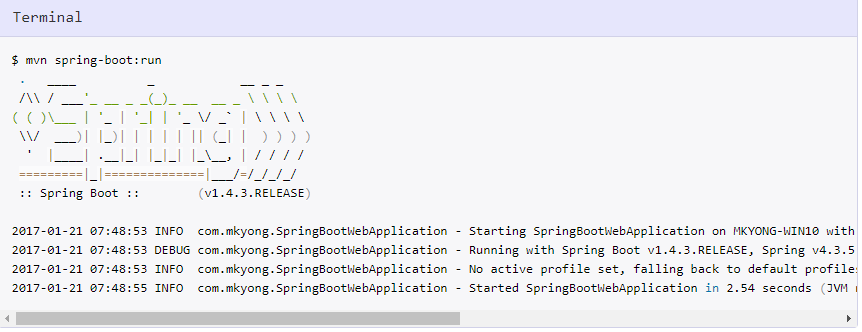
By default, Spring Boot max file upload size is 1MB, you can configure the values via following external configuration file application properties :

application.properties

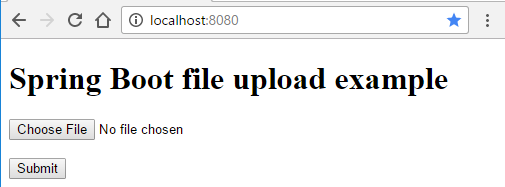
|  |
| --- |
| spring.http.multipart.max-file-size=10MB  spring.http.multipart.max-request-size=10MB |

Testing

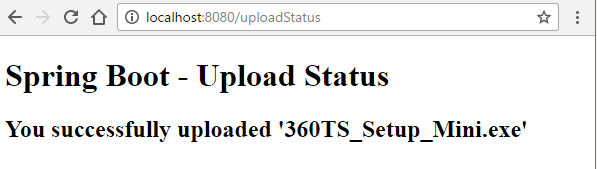
Start Spring Boot with the default embedded Tomcat mvn spring-boot:run.



 Access <http://localhost:8080/>



7.2 Select a file and upload it.



7.3 Select a file larger than 10mb, you will visit this page.

