

Follow @only2dhir

Uploading and Downloading Files with Spring Boot



By [Dhiraj](#), 19 August, 2019

135K



In this tutorial, we will learn different ways with which we can upload and download files such as pdf, .zip file or images with [spring boot](#) and REST. The implementation will have examples to upload and download single and multiple files. While uploading, we will have choices to either save the uploaded file in the local file system with [Resource](#) provided in Spring framework or save it to the database. We will be using MySQL for this quick tutorial.

While downloading multiple files, we will also have an implementation to zip all the files in a single unit and then download it from the spring boot server. We will also look into how to send extra params with form data while uploading the files.

At the end, we will test our example with Postman.

Spring Boot Project Setup

Head over to <https://start.spring.io> to generate our spring boot demo project with below artifacts.

Selected dependencies

Spring Web Starter Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.	✓
Spring Data JPA Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.	✓
MySQL Driver MySQL JDBC driver.	✓

Below is our pom.xml for those who have generated their project already.

Recommended

File Upload React Spring Rest
File Upload Angularjs Spring Boot Rest
Spring Boot Multiple Database Configuration
Spring Boot H2 Database Example
Spring Data Jpa Example
Spring Boot Jms Activemq Example



```
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Spring Boot File Upload Configuration

Below are the multipart configurations required in application.properties to enable file uploading in a Spring Boot app.

```
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=15MB
```

max-file-size – It specifies the maximum size permitted for uploaded files. The default is 1MB.

max-request-size – It specifies the maximum size allowed for multipart/form-data requests. The default is 10MB.

spring.servlet.multipart.enabled – Whether to enable support of multipart uploads.

Spring Boot File Upload

In this section, we will provide the different options of uploading the files in a spring boot app with suitable examples.

Single File Upload to Local File System in Spring Boot Rest

file received in a multipart request.

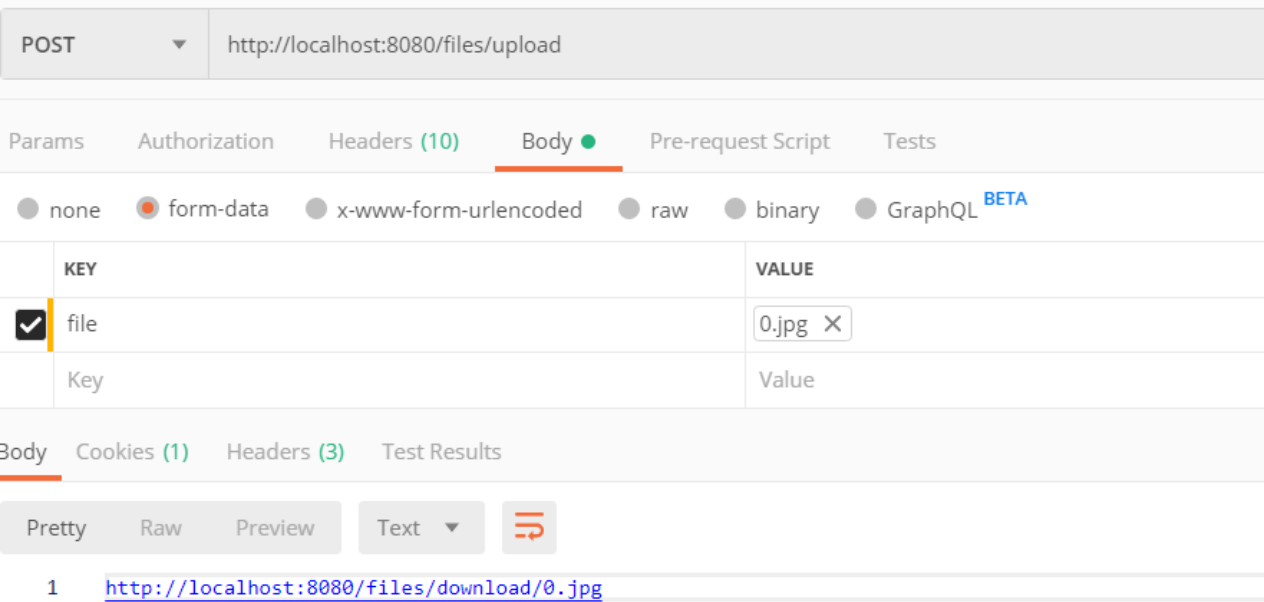


In the below implementation, we are Copying all bytes from an input stream to a file. By default, the copy fails if the target file already exists or is a symbolic link. Hence, we are using standard copy option as [REPLACE_EXISTING](#).

Once, this process is completed, the response will be the download URL of the file. We will discuss the file download in a moment.

```
@PostMapping("/upload")
public ResponseEntity uploadToLocalFileSystem(@RequestParam("file") MultipartFile file) {
    String fileName = StringUtils.cleanPath(file.getOriginalFilename());
    Path path = Paths.get(fileBasePath + fileName);
    try {
        Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE_EXISTING);
    } catch (IOException e) {
        e.printStackTrace();
    }
    String fileDownloadUri = ServletUriComponentsBuilder.fromCurrentContextPath()
        .path("/files/download/")
        .path(fileName)
        .toUriString();
    return ResponseEntity.ok(fileDownloadUri);
}
```

Below is the sample request that we can make to test this functionality from Postman. You can also use javascript or any other JS library to test it.



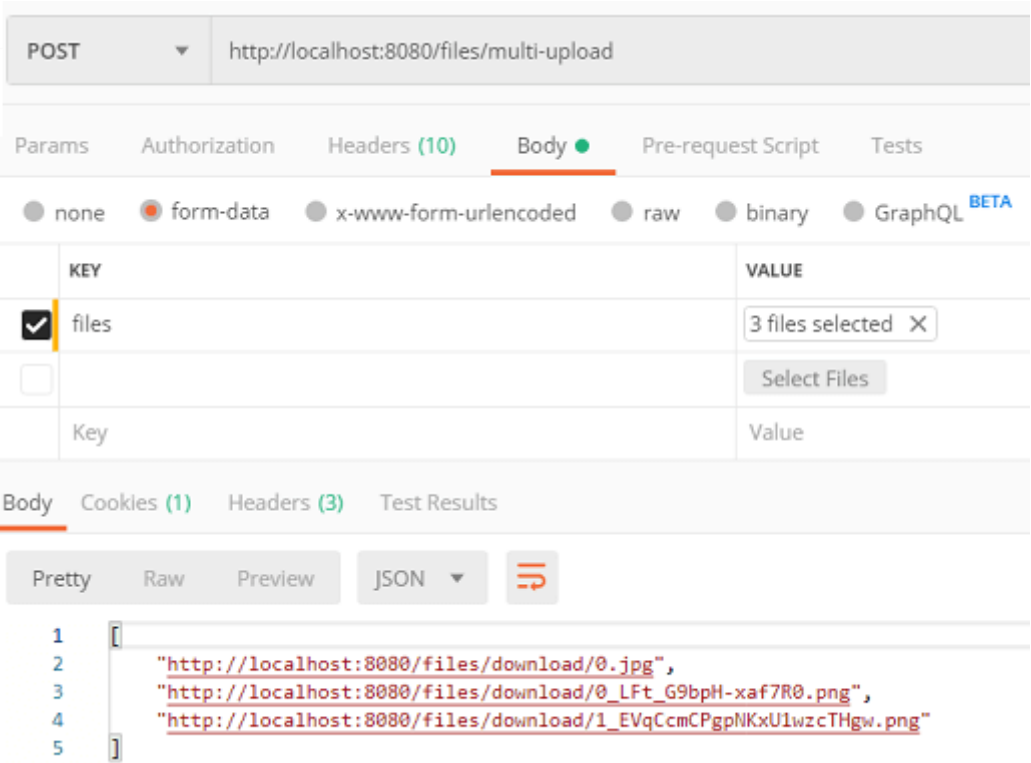
Multiple File Uploads to Local File System in Spring Boot Rest

The multiple files upload internally invokes above method to repeat the single file upload. We can also have a functionality to upload a .zip file from the client and in the server-side, we can unzip it and save it in our local file system individually.



```
        List<Object> fileDownloadUrIs = new ArrayList<>();
        Arrays.asList(files)
            .stream()
            .forEach(file -> fileDownloadUrIs.add(uploadToLocalFileSystem
(file).getBody()));
        return ResponseEntity.ok(fileDownloadUrIs);
    }
```

Below is the screenshot of Postman to test this multiple file upload.



Adding Extra Parameters with FileUpload

To add an extra parameter with file upload, we can append that extra parameter in the form data at the client-side and the same can be retrieved as a request param at the server-side. Below is an example where we appended the key as `extraParam` in the form data at client-side.

```
@PostMapping("/upload-extra-param")
public ResponseEntity uploadWithExtraParams(@RequestParam("file") MultipartFile file,
@RequestParam String extraParam) {
    logger.info("Extra param " + extraParam);
    return uploadToLocalFileSystem(file);
}
```

`uploadToLocalFileSystem()` is the same method that we defined above.

Spring Boot File Upload to Database

For this, we need to have a database configuration first. Spring boot provides a very convenient way to do so by adding a few properties in `application.properties`. We will be using [spring data JPA](#) for our purpose.

`application.properties`

```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
spring.user.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```



To save the uploaded file in the DB, we have a model class and we are using `byte[]` as a data type to save it in the DB.

Document.java

```
@Entity
public class Document {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column
    private String docName;

    @Column
    @Lob
    private byte[] file;

}
```

Below is the REST implementation for this. The implementation is similar to above logic except it has a DB call to save the file instead of saving it to the local file system.

```
@PostMapping("/upload/db")
public ResponseEntity uploadToDB(@RequestParam("file") MultipartFile file) {
    Document doc = new Document();
    String fileName = StringUtils.cleanPath(file.getOriginalFilename());
    doc.setDocName(fileName);
    try {
        doc.setFile(file.getBytes());
    } catch (IOException e) {
        e.printStackTrace();
    }
    documentDao.save(doc);
    String fileDownloadUri = ServletUriComponentsBuilder.fromCurrentContextPath()
        .path("/files/download/")
        .path(fileName).path("/db")
        .toUriString();
    return ResponseEntity.ok(fileDownloadUri);
}
```

File Download in Spring Boot

Spring Boot File Download from Local File System

You have already noticed the response of the file upload. It is a simple GET URL and on the click of that URL the file will be downloaded automatically in the browser as we will be adding [Content-Disposition](#) in the response header as an attachment and the content type as [application/octet-stream](#).



```
@GetMapping("/download/{fileName:.+}")
public ResponseEntity downloadFileFromLocal(@PathVariable String fileName) {
    Path path = Paths.get(fileBasePath + fileName);
    Resource resource = null;
    try {
        resource = new UrlResource(path.toUri());
    } catch (MalformedURLException e) {
        e.printStackTrace();
    }
    return ResponseEntity.ok()
        .contentType(MediaType.parseMediaType(contentType))
        .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename"
= "\" + resource.getFilename() + "\"")
        .body(resource);
}
```

Spring Boot File Download from Database

The below implementation only difers in the process of getting the file from databse rather than a file system.

```
@GetMapping("/download/{fileName:.+}/db")
public ResponseEntity downloadFromDB(@PathVariable String fileName) {
    Document document = documentDao.findByDocName(fileName);
    return ResponseEntity.ok()
        .contentType(MediaType.parseMediaType(contentType))
        .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename"
= "\" + fileName + "\"")
        .body(document.getFile());
}
```

Zip Multiple Files and Download

While downloading multiple files, we can create a zip file in spring boot and download that zip file alone rather than downloading multiple files individually. For this purpose, we first need to create a zip file in spring boot and then set the content type as application/zip to download the zip file.

Here, we will be using ZipOutputStream from java.util.zip package to create the .zip file. Below is the example.



```
ZipOutputStream zipOut = new ZipOutputStream(response.getOutputStream());
for (String fileName : name) {
    FileSystemResource resource = new FileSystemResource(fileBasePath + fi
leName);

    ZipEntry zipEntry = new ZipEntry(resource.getFilename());
    zipEntry.setSize(resource.contentLength());
    zipOut.putNextEntry(zipEntry);
    StreamUtils.copy(resource.getInputStream(), zipOut);
    zipOut.closeEntry();
}
zipOut.finish();
zipOut.close();
response.setStatus(HttpServletResponse.SC_OK);
response.addHeader(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\""
+ zipFileName + "\"");
}
```

To test this functionality, we can make a GET request with name as a query parameter which includes all the file names that we want to be included in the final .zip file.

```
http://localhost:8080/files/zip-download?name=1.png&name=0.jpg
```

Spring Boot File Upload with React and Angular

We tested the above implementation with Postman. But in my previous tutorial, we have created other tutorials to achieve the same functionality with popular Javascript framework and library.

You can follow [this article](#) to implement the file uploading client with Angular and [this tutorial with React Js](#).

Conclusion

In this article, we disussed about different ways to upload and download files and images with spring boot REST and tested with Postman.


If You Appreciate This, You Can Consider:



Like us at: or follow us at

Share this article on social media or with your teammates.

About The Author



A technology savvy professional with an exceptional capacity to analyze, solve problems and multi-task. Technical expertise in highly scalable distributed systems, self-healing systems, and service-oriented architecture.

Kibana, Elasticsearch, etc.

Further Reading on Spring Boot

- 1. [File Upload React Spring Rest](#)
- 2. [File Upload Angularjs Spring Boot Rest](#)
- 3. [Spring Boot Multiple Database Configuration](#)
- 4. [Spring Boot H2 Database Example](#)
- 5. [Spring Data Jpa Example](#)
- 6. [Spring Boot Jms Activemq Example](#)



ALSO ON DEVGLAN

Spring Boot Admin

a year ago • 1 comment

In this article, we will discuss Spring Boot Admin by utilizing actuator ...

Spring Boot MongoDB Configuration

2 years ago • 3 comments

In this quick article, we will deal with spring boot mongo DB configuration. Spring ...

RSA Encryption in Javascript and ...

2 years ago • 3 comments

In this article, we will learn about RSA encryption and how to perform RSA ...

What do you think?

18 Responses



Upvote



Funny



Love



Surprised



Angry



Sad

4 Comments

Devglan

🔒 Disqus' Privacy Policy

1 Login ▾

❤ Recommend

🐦 Tweet

📌 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name



mitrich • a year ago

What is fileBasePath?

1 ^ | ▾ • Reply • Share ›



Dovi ➔ mitrich • a year ago

A String variable containing the path where the files are stored.

1 ^ | ▾ • Reply • Share ›



Ashvini Mishra • 9 months ago • edited