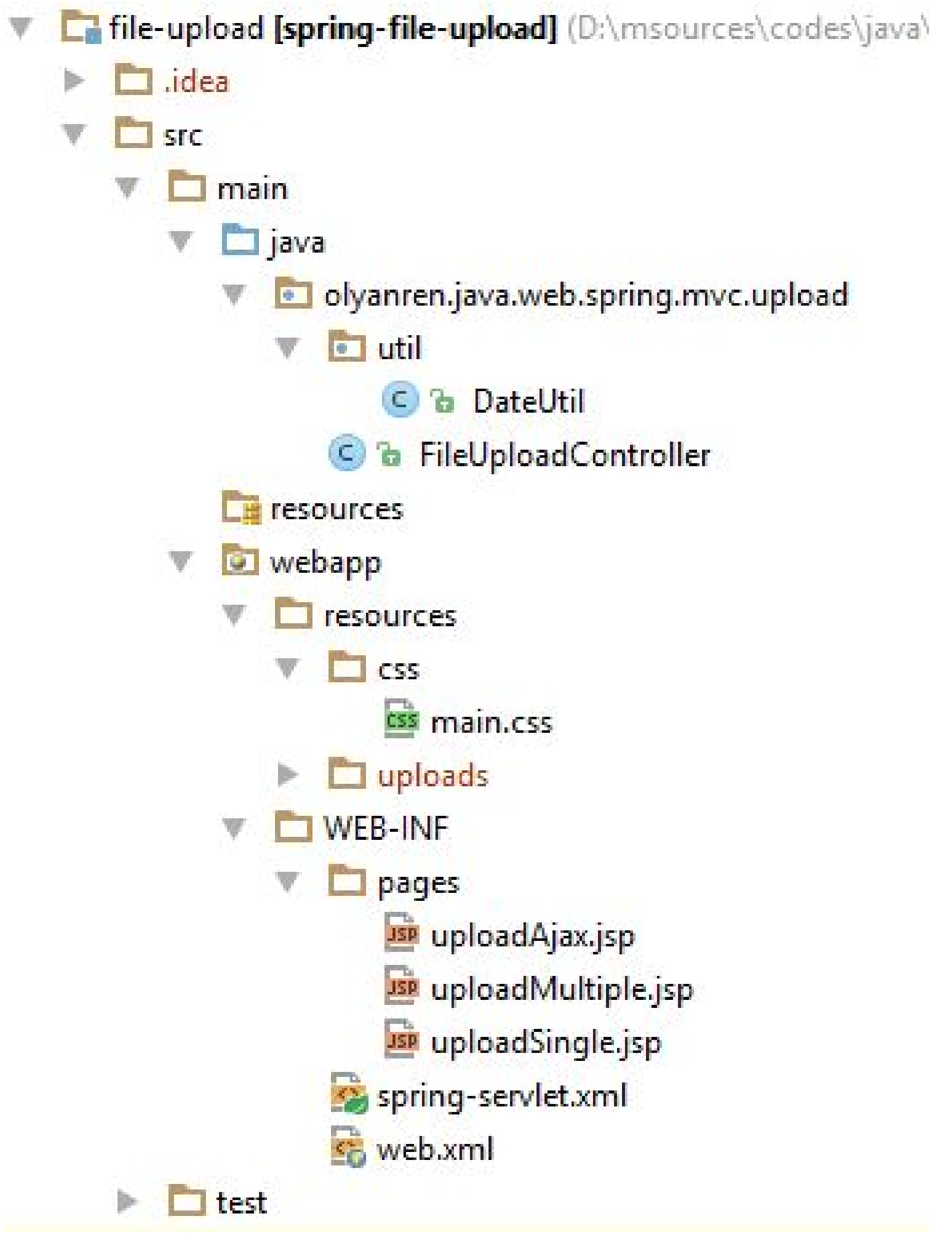
Spring MVC File Upload

File Uploading is a very common in any web application. In this tutorial, we will introduce three different file upload operations: single file upload, multipe file upload, and ajax based file upload.

First of all, there are some pre-request steps needed:

1. Create a Maven project
2. Add Spring Framework dependencies
3. Add Javax Servlet and JSTL dependencies
3. Add Apache Commons dependencies

After initialization web application, directory structure of the application will look below image:



```
file-upload [spring-file-upload] (D:\msources\codes\java\
    .idea
    src
        main
            java
                olyanren.java.web.spring.mvc.upload
                    util
                        DateUtil
                    FileUploadController
            resources
            webapp
                resources
                    css
                        main.css
                    uploads
                WEB-INF
                    pages
                        uploadAjax.jsp
                        uploadMultiple.jsp
                        uploadSingle.jsp
                    spring-servlet.xml
                    web.xml
        test
```

Necessary Dependencies

As stated above, we need following dependencies:

Spring Framework Dependencies

```xml
<!--Spring MVC dependencies-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
</dependency>
```

Javax Servlet And JSTL Dependencies

```xml
<!--Javax Servlet-->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
<!-- jstl -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

Apache Commons Dependencies

```xml
<!-- Apache Commons FileUpload -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>
<!-- Apache Commons IO -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
</dependency>
```

JSP Pages

We will create three JSP pages to allow single, multiple and ajax-based file uploads.

uploadSingle.jsp

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page session="false" %>
<html>
<head>
    <title>Classic Single File Upload</title>
    <style type="text/css">
        ul li {
            list-style: none;
            float: left;
        }
    </style>
</head>
<body>
<form method="POST" action="<c:url value="/upload-single" />" enctype="multipart/form-data">
    <ul>
        <li>
            <input type="file" name="file"/>
        </li>
        <li>
            <input type="submit" style="margin-left: 20px" value="Upload">
        </li>
        <li>
            <span id="fileNames"></span>
        </li>
    </ul>
</form>
</body>
</html>
```

uploadMultiple.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page session="false" %>
<html>
<head>
    <title>Classic Multiple File Upload</title>
    <style type="text/css">
        ul li {
            list-style: none;
            float: left;
        }
    </style>
</head>
<body>
<form method="POST" action="<c:url value="/upload-multiple" />" enctype="multipart/form-data">
    <ul>
        <li>
            <div>
                <input type="file" name="file[]" multiple/>
                <input type="submit" style="margin-left: 20px" value="Upload">
            </div>
            <div style="clear: both"></div>
        </li>
        <li>
            <span id="fileNames"></span>
        </li>
    </ul>
</form>
</body>
</html>
```

uploadAjax.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" %>
<%@ page session="false" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<!DOCTYPE html>
<html>
<head>
        <link href="<c:url value="/resources/css/main.css" />" rel="stylesheet">
    <title>File Upload With Ajax</title>
    <script type="text/javascript">

        $(document).ready(function () {
            $('progress').hide();
            $('.trigger-file-input').click(function () {
                $('#file').click();
            });
            $("input:file").change(function () {
                $('progress').show();
                var files = document.getElementById('file').files;
                var formData = new FormData();
                for (var i = 0; i < files.length; i++) {
                    formData.append("file" + i, files[i]);
                }
                $.ajax({
                    url: "/upload-ajax", //Server script to process data
                    type: 'POST',
                    xhr: function () { // Custom XMLHttpRequest
                        var myXhr = $.ajaxSettings.xhr();
                        if (myXhr.upload) {
                            myXhr.upload.addEventListener('progress', progressHandlingFunction,
false);
                        }
                        return myXhr;
                    },
                    //Ajax events
                    beforeSend: beforeSendHandler,

                    error: errorHandler,
                    // Form data
                    data: formData,
                    //Options to tell jQuery not to process data or worry about content-type.
                    cache: false,
                    contentType: false,
                    processData: false,
                    success: function (e) {
                        $("#result").html(e);
                    }

                });
            });

            function beforeSendHandler() {
            }

            function errorHandler(e) {
                alert("An error occurred");
            }

            function progressHandlingFunction(e) {
                if (e.lengthComputable) {
                    $('progress').attr({value: e.loaded, max: e.total});
                }
            }
        });
    </script>
</head>
<body>
<div>
    <ul>
        <li>
            <div >
                <button type="button" class="buttonSubmit trigger-file-input">Choose
Files</button>
            </div>
        </li>
        <li>
            <input type="file" id="file" multiple name="file[]">
        </li>
        <li>
            <progress></progress>
        </li>
    </ul>
    <div id="result">
    </div>
</div>
</body>
</html>
```

Notice that these files except ajax based file contain form element with `encytype=multipart/form-data`, and `<input type="file">` element. To handle files, we should give a name to the file input element, so these jsp pages contain `<input type="file">` named as `"file"` and `"file[]"`

Spring Configuration

In Spring configuration, we register Apache Commons FileUpload for handling multipart requests.

spring-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:beans="http://www.springframework.org/schema/beans"
            xmlns:context="http://www.springframework.org/schema/context"
            xmlns:mvc="http://www.springframework.org/schema/mvc"
            xsi:schemaLocation="http://www.springframework.org/schema/mvc
             http://www.springframework.org/schema/mvc/spring-mvc.xsd
             http://www.springframework.org/schema/beans
             http://www.springframework.org/schema/beans/spring-beans.xsd
             http://www.springframework.org/schema/context
             http://www.springframework.org/schema/context/spring-context.xsd">


    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven/>
    <context:component-scan base-package="olyanren"/>
    <mvc:resources mapping="/resources/**" location="/resources/" />
    <!-- Resolves views selected for rendering by @Controllers to .jsp resources
    in the /WEB-INF/pages directory -->
    <beans:bean
       class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/pages/"/>
        <beans:property name="suffix" value=".jsp"/>
    </beans:bean>

    <beans:bean id="multipartResolver"
                class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <beans:property name="maxUploadSize" value="1024000"/>
    </beans:bean>
</beans:beans>
```

Notice that we configured **multipartResolver** bean by assigning **CommonsMultipartResolver** class. The id should be **filterMultipartResolver** if you are using Spring security, so when you change this id value, spring cannot upload files.

Also note that we have configured maximum upload size limit by providing **maxUploadSize** property. This number specifies byte size.

Note: If you are using Spring security, you should add multipart filter before spring security filter chain as follows:

```
<filter>
    <display-name>springMultipartFilter</display-name>
    <filter-name>springMultipartFilter</filter-name>
    <filter-class>org.springframework.web.multipart.support.MultipartFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>springMultipartFilter</filter-name>
    <url-pattern>/*</url-pattern>
```

```xml
</filter-mapping>
<!-- Spring Security -->
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Spring Controller Class

```java
@Controller
public class FileUploadController {
     private static String FILE_SEPARATOR = "/"

     @RequestMapping(value ="/upload-multiple", method = RequestMethod.GET)
     public String uploadMultiple(){
         return "uploadMultiple"
     }
     @RequestMapping(value ="/upload-single", method = RequestMethod.GET)
     public String upload(){
         return "uploadSingle"
     }
     @RequestMapping(value ="/upload-ajax", method = RequestMethod.GET)
     public String uploadAjax(){
         return "uploadAjax"
     }
     @RequestMapping(value = "/upload-multiple", method = RequestMethod.POST,
             produces = "text/html;charset=UTF-8")
     @ResponseBody
     public String uploadMultipleFileHandler(
             @RequestParam("file[]") MultipartFile[] files, HttpServletRequest request) {

         String storedFolderLocation = createStoredFolder(request);
         String fileSeparator = "/";
         String urls = "";

         for (MultipartFile file : files) {
             String uploadedFileName = file.getOriginalFilename();
             try {
                 byte[] bytes = file.getBytes();

                 String storedFileLocation = storedFolderLocation + fileSeparator +
uploadedFileName;
                 // Create the file on server
                 File serverFile = new File(storedFileLocation);
                 BufferedOutputStream stream = new BufferedOutputStream(
                         new FileOutputStream(serverFile));
                 stream.write(bytes);
                 stream.close();


                 String url = getDomainName(request)
                         + getRelativePath() + fileSeparator + uploadedFileName;
                 if (isFileTypeImage(uploadedFileName)) {
                     urls += "<img src=\"" + url + "\" />";
                 } else {
                     urls += "<a href=\"" + url + "\">" + url + "</a>";
                 }

             } catch (Exception e) {
                 return "You failed to upload " + uploadedFileName + " => " + e.getMessage();
             }
```

```java
        }
        return "Loaded Files:"+urls;
    }
    @RequestMapping(value = "/upload-single", method = RequestMethod.POST,
            produces = "text/html;charset=UTF-8")
    @ResponseBody
    public String uploadFileHandler(
            @RequestParam("file") MultipartFile file, HttpServletRequest request) {
        String url;
        String storedFolderLocation = createStoredFolder(request);
            String uploadedFileName = file.getOriginalFilename();
            try {
                byte[] bytes = file.getBytes();

                String storedFileLocation = storedFolderLocation + FILE_SEPARATOR +
uploadedFileName;
                // Create the file on server
                File serverFile = new File(storedFileLocation);
                BufferedOutputStream stream = new BufferedOutputStream(
                        new FileOutputStream(serverFile));
                stream.write(bytes);
                stream.close();
                url = getDomainName(request)
                        + getRelativePath() + FILE_SEPARATOR + uploadedFileName;
                if (isFileTypeImage(uploadedFileName)) {
                   url= "<img src=\"" + url + "\" />";
                } else {
                    url= "<a href=\"" + url + "\">" + url + "</a>";
                }

            } catch (Exception e) {
                return e.getMessage();
            }
        return "Loaded File:"+url;
    }
    @RequestMapping(value = "upload-ajax", method = RequestMethod.POST)
    @ResponseBody
    public String uploadMultipleFiles(MultipartHttpServletRequest request) {
        CommonsMultipartFile multipartFile = null;
        Iterator<String> iterator = request.getFileNames();
        String filePaths = "";
        while (iterator.hasNext()) {
            String key = iterator.next();
            // create multipartFile array if you upload multiple files
            multipartFile = (CommonsMultipartFile) request.getFile(key);
            String uploadedFileName = multipartFile.getOriginalFilename();
            try {
                byte[] bytes = multipartFile.getBytes();

                String storedFileLocation = createStoredFolder(request) + FILE_SEPARATOR +
uploadedFileName;
                // Create the file on server
                File serverFile = new File(storedFileLocation);
                BufferedOutputStream stream = new BufferedOutputStream(
                        new FileOutputStream(serverFile));
                stream.write(bytes);
                stream.close();

                String url = getDomainName(request)
                        + getRelativePath() + FILE_SEPARATOR + uploadedFileName;
                if (isFileTypeImage(uploadedFileName)) {
                    filePaths += "<img src=\"" + url + "\" />";
                } else {
                    filePaths += "<a href=\"" + url + "\">" + url + "</a>";
                }


                } catch (Exception e) {
                    return e.getMessage();
                }
            }

            return filePaths;
        }
        private String createStoredFolder(HttpServletRequest request) {
            String realPath = request.getSession().getServletContext().getRealPath("/");
            String relativePath = getRelativePath();
            String storedFolderLocation = realPath + relativePath;
            File dir = new File(storedFolderLocation);
            if (!dir.exists()) {
                dir.mkdirs();
            }
            return storedFolderLocation;
        }
        private boolean isFileTypeImage(String fileName) {
            String imagePattern =
                    "([^\\s]+(\\.(?i)(jpg|jpeg|png|gif|bmp))$)";
            return Pattern.compile(imagePattern).matcher(fileName).matches();

        }
        private String getRelativePath() {
            String fileSeparator = "/";
            DateUtil dateUtil = new DateUtil();
            int[] yearMonthDay = dateUtil.getDayMonthYear();
            return "/resources/uploads/" + yearMonthDay[0] + fileSeparator
                    + yearMonthDay[1] + fileSeparator + yearMonthDay[2];
        }
        private String getDomainName(HttpServletRequest request) {
            return request.getProtocol().toLowerCase().replaceAll("[0-9./]", "") + "://" +
                    request.getServerName() + ":" + request.getServerPort();
        }
}
```