# Subida de archivos con Spring

Lo primero que hemos necesitado es actualizar el servlet de Spring a la versión 3.1.0, para poder utilizar la interfaz de MultipartResolver, añadiendo la dependencia en el pom.xml, además la de “commons-fileupload”. Además, hay que añadir el “bean” en el servlet.xml.

```
331
332         <dependency>
333             <groupId>commons-fileupload</groupId>
334             <artifactId>commons-fileupload</artifactId>
335             <version>1.3.1</version>
336         </dependency>
337
266
267         <dependency>
268             <groupId>javax.servlet</groupId>
269             <artifactId>javax.servlet-api</artifactId>
270             <version>3.1.0</version>
271             <scope>provided</scope>
272         </dependency>
```

```
122     <bean id="viewResolver"
123         class="org.springframework.web.servlet.view.UrlBasedViewResolver">
124         <property name="viewClass"
125             value="org.springframework.web.servlet.view.tiles2.TilesView" />
126     </bean>
127
128
129     <bean class="org.springframework.web.multipart.commons.CommonsMultipartResolver" id="multipartResolver"/>
```

Una vez hecho esto, creamos dos clases java para configurar el MultipartResolver.

```
4  import java.io.File;
10
11 public class MyWebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
12
13     private final int   maxUploadSizeInMb   = 5 * 1024 * 1024;  // 5 MB
14
15
16     @Override
17     protected Class<?>[] getServletConfigClasses() {
18         return new Class[] {
19             SpringWebMvcConfig.class
20         };
21     }
22
23     @Override
24     protected String[] getServletMappings() {
25         return new String[] {
26             "/"
27         };
28     }
```

```
29
30     @Override
31     protected Class<?>[] getRootConfigClasses() {
32         return null;
33     }
34
35     @Override
36     protected void customizeRegistration(final ServletRegistration.Dynamic registration) {
37
38         // upload temp file will put here
39         final File uploadDirectory = new File(System.getProperty("java.io.tmpdir"));
40
41         // register a MultipartConfigElement
42         final MultipartConfigElement multipartConfigElement = new MultipartConfigElement
43             (uploadDirectory.getAbsolutePath(), this.maxUploadSizeInMb,
44                 this.maxUploadSizeInMb * 2, this.maxUploadSizeInMb / 2);
45
46         registration.setMultipartConfig(multipartConfigElement);
47
48     }
49
50 }
```

```java
package com;

import org.springframework.context.annotation.Bean;

@EnableWebMvc
@Configuration
@ComponentScan({
    "com"
})
public class SpringWebMvcConfig extends WebMvcConfigurerAdapter {

    // Bean name must be "multipartResolver", by default Spring uses method name as bean name.
    @Bean
    public MultipartResolver getMultipartResolver() {
        return new StandardServletMultipartResolver();
    }

    /*
     * // if the method name is different, you must define the bean name manually like this :
     *
     * @Bean(name = "multipartResolver")
     * public MultipartResolver createMultipartResolver() {
     * return new StandardServletMultipartResolver();
     * }
     */

    @Bean
    public InternalResourceViewResolver viewResolver() {
        final InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-INF/views/jsp/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }

}
```

Creamos dos vistas, una para seleccionar el archivo a subir y otra para enseñar un mensaje cuando se suba.

```
<html>


<body>


<form method="POST" action="upload/upload.do" enctype="multipart/form-data">
   <input type="file" name="file" /><br/>
   <input type="submit" value="Submit" />
</form>


</body>
</html>
```

```
<html>

<body>

<h2>Message : ${message}</h2>

</body>

</html>
```

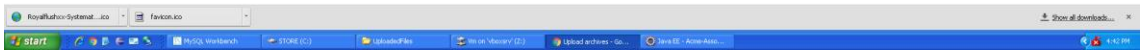
Creamos un controlador para gestionar esto:

```
package controllers;

import java.io.IOException;

@Controller
@RequestMapping("/upload")
public class UploadController extends AbstractController {

    //Save the uploaded file to this folder
    private static String   UPLOADED_FOLDER = "C://UploadedFiles//";


    @RequestMapping(value = "/subir", method = RequestMethod.GET)
    public ModelAndView index() {
        ModelAndView result;
        result = new ModelAndView("upload/upload");
        return result;
    }
```

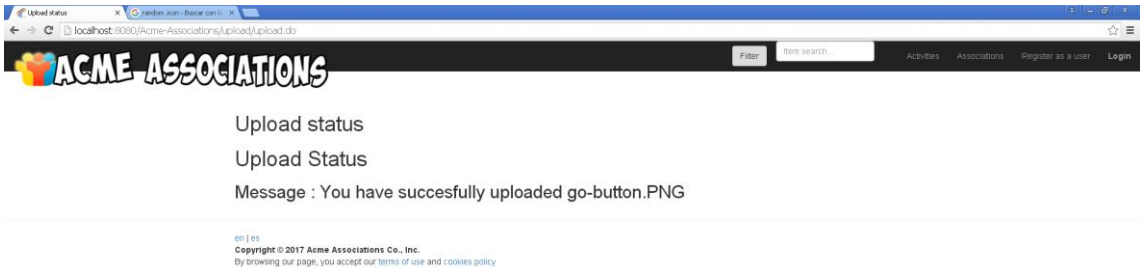
```
    @RequestMapping(value = "/upload", method = RequestMethod.POST)
    public ModelAndView singleFileUpload(@RequestParam("file") final MultipartFile file, final RedirectAttributes redirectAttributes) {

        final ModelAndView result;
        String message = null;

        if (file.isEmpty()) {

            result = new ModelAndView("upload/subir");
            redirectAttributes.addFlashAttribute("message", "Please select a file to upload");
            return result;
        }

        try {

            // Get the file and save it somewhere
            final byte[] bytes = file.getBytes();
            final Path path = Paths.get(UploadController.UPLOADED_FOLDER + file.getOriginalFilename());
            Files.write(path, bytes);

            message = "You successfully uploaded '" + file.getOriginalFilename() + "'";

        } catch (final IOException e) {
            e.printStackTrace();
        }

        result = new ModelAndView("upload/status");
        result.addObject("message", message);
        return result;
    }
```

```
    @RequestMapping(value = "/uploadStatus", method = RequestMethod.GET)
    public ModelAndView uploadStatus() {
        ModelAndView result;
        result = new ModelAndView("upload/status");
        return result;
    }

    @RequestMapping(value = "/uploadMultiPage", method = RequestMethod.GET)
    public String uploadMultiPage() {
        return "uploadMulti";
    }

}
```

Una vez añadido todo esto, forzamos la actualización de Maven e iniciamos el servidor.

Seleccionamos un archivo y le damos a Submit.




El archivo se encontrará en la carpeta “C:\UploadedFiles”, que debemos haber creado previamente.

En caso de no seleccionar un archivo, se mostrará un mensaje de error.