文件上传

其实难度不大

先放页面

```
<h3>文件上传</h3>
<form action="upload" enctype="multipart/form-data" method="post">
    <table>
        <tr>
            <td>文件描述: </td>
            <td><input type="text" name="description"></td>
        </tr>
        <tr>
            <td>请选择文件: </td>
            <td><input type="file" name="file"></td>
        </tr>
        <tr>
            <td><input type="submit" value="上传"></td>
        </tr>
    </table>
</form>
```

因为是该表单要上传表单，所以必须设置enctype=“mulipart/form-data”

在springmvc的配置文件中必须加这一段

```
<!-- 多部分文件上传 -->
<beans:bean id="multipartResolver"
    class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <beans:property name="maxUploadSize" value="104857600" />
    <beans:property name="maxInMemorySize" value="4096" />
    <beans:property name="defaultEncoding" value="UTF-8" />
</beans:bean>
```

标签beans:bean还是bean视自己的配置文件而定

控制器

```
@RequestMapping(value = "/upload", method = RequestMethod.POST)
public String upload(HttpServletRequest request, @RequestParam String description,
        @RequestParam MultipartFile file)
        throws Exception {
    if (!file.isEmpty()) {
        // 上传文件的路径,指定了服务器（Tomcat）下该项目根目录下一个文件夹名为images的文件夹里
        String path = request.getServletContext().getRealPath("/images/");
        // 获取了上传文件的文件名
        String filename = file.getOriginalFilename();
        // 根据路径和文件名新建一个文件
        File filepath = new File(path, filename);
        // 确保该文件的上级目录存在，不存在就创建
        if (!filepath.getParentFile().exists()) {
            filepath.getParentFile().mkdirs();
        }
        // 将文件写入
        file.transferTo(filepath);
        return "success";
    } else {
        return "error";
    }
}
```

springmvc会将上传文件绑定到MulipartFile对象中，MulipartFile提供了获取文件文件内容、文件名，通过transferTo方法将文件存储到硬件中

上传成功后我们部署在tomcat下的项目的根目录下文件夹images下就会有我们的文件了

注意：因为文件是上传到服务器的，而我们ide每次将项目部署到服务器时都是将WebContent重新部署一下，所以会出现我修改了东西后，再去服务器查看时上传文件已经没了，因为可以看作项目已经重新部署了一次，你项目里本身存在什么，不存在什么，服务器下也是存在什么，不存在什么

接下来我们试试用对象接收上传文件，其实就是将两个属性封装成一个持久化类

持久化类

```
public class UserImage implements Serializable {

    private String username;
    private MultipartFile image;
```

上传界面

```
<h3>用户注册</h3>
<form action="register" enctype="multipart/form-data" method="post">
    <table>
        <tr>
            <td>用户名: </td>
            <td><input type="text" name="username"></td>
        </tr>
        <tr>
            <td>请上传头像: </td>
            <td><input type="file" name="image"></td>
        </tr>
        <tr>
            <td><input type="submit" value="注册"></td>
        </tr>
    </table>
</form>
```

控制器方法

```
@RequestMapping(value = "/register")
public String register(HttpServletRequest request,
        @ModelAttribute UserImage user, Model model) throws Exception {
    System.out.println(user.getUsername());
    if (!user.getImage().isEmpty()) {
        String path = request.getServletContext().getRealPath("/images/");
        String filename = user.getImage().getOriginalFilename();
        File filepath = new File(path, filename);
        if (!filepath.getParentFile().exists()) {
            filepath.getParentFile().mkdirs();
        }
        user.getImage().transferTo(filepath);
        model.addAttribute("user", user);
        return "userInfo";
    } else {
        return "error";
    }
}
```

其实看着和之前的也没什么差别，只不过传过来的两个属性都存在一个持久化类里而已

接下来既然我们考虑了上传 肯定得有下载吧

下载页面

```
<h3>文件下载</h3>
<a
    href="download?filename=${requestScope.user.image.originalFilename }">
    ${requestScope.user.image.originalFilename } </a>
```

控制器方法

```
@RequestMapping(value = "/download")
public ResponseEntity<byte[]> download(HttpServletRequest request,
        @RequestParam("filename") String filename,
        Model model) throws Exception {
    // 下载文件的路径
    String path = request.getServletContext().getRealPath("/images/");
    File file = new File(path + File.separator + filename);
    HttpHeaders headers = new HttpHeaders();
    // 下载显示的文件名，解决中文乱码问题
    String downloadFileName = new String(filename.getBytes("UTF-8"), "iso-8859-1");
    // 通知浏览器以attachment（下载方式）打开图片
    headers.setContentDispositionFormData("attachment", downloadFileName);
    // 二进制流数据
    headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);

    return new ResponseEntity<byte[]>(FileUtils.readFileToByteArray(file), headers,
            HttpStatus.CREATED);
}
```

表示只想拿过来用