此章节我们关注学习两种文件上传方式：Multipartfile方式文件上传和ServletFileUpload方式文件上传。

两种方式给出代码案例，从不同侧来学习Java中文件上传方式。

以下给出的文件上传代码均为示例代码，无安全防护，目前仅是为了让大家学习理解这几种上传方式。

提前铺垫下一阶段任意文件上传漏洞前置知识。

当然还有其他方式和组件的文件上传，比如文件流方式，smartupload组件等等。

本次仅讲述常见的两种。不论用那种方式都换汤不换药，方式的不同对于我们后面进行任意文件上传代码审计影响不会太大。到时再具体问题具体分析。

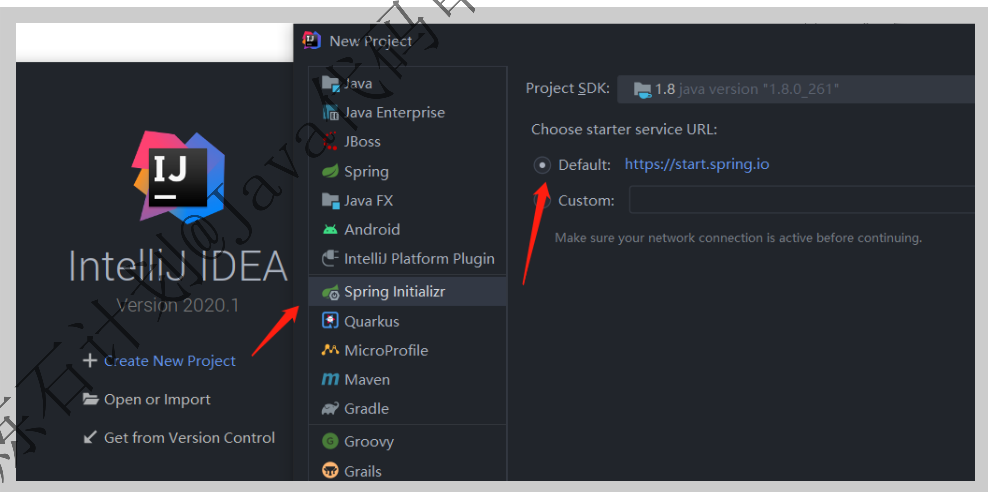
一、Multipartfile方式文件上传

1、简介

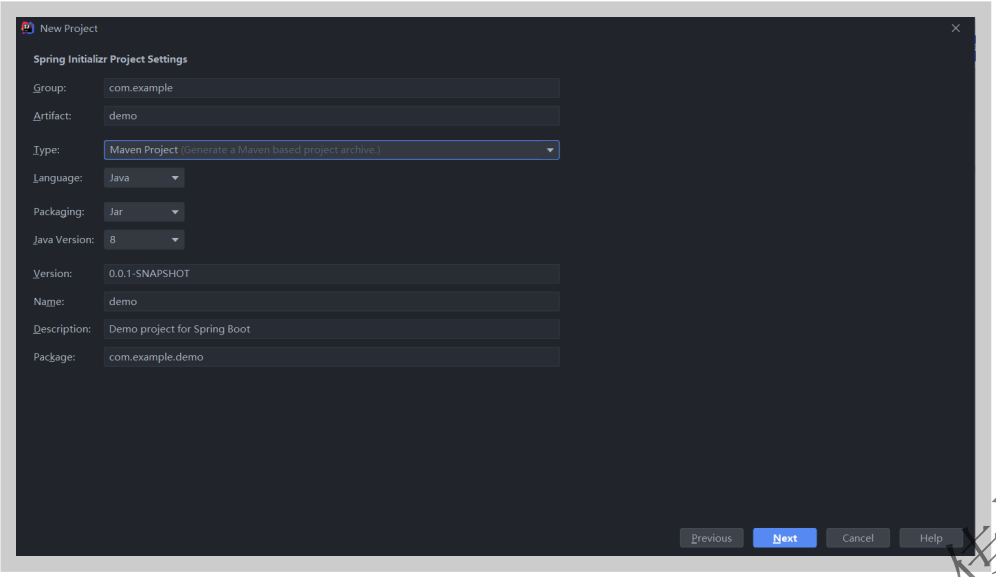
MultipartFile是SpringMVC提供简化上传操作的工具类。

2、环境搭建

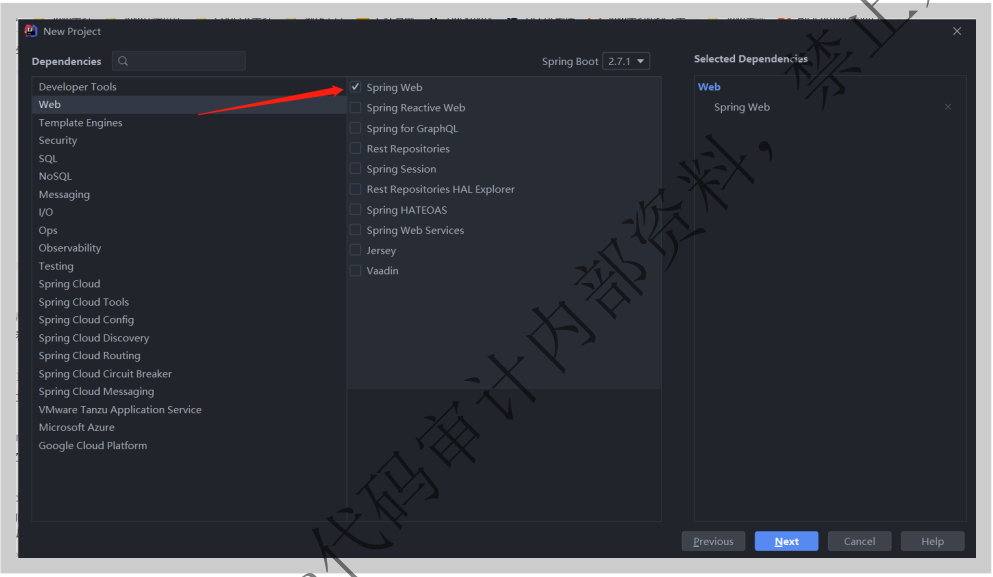
①、创建SpringBoot项目，打开IDEA，点击Create New Project，选择Spring Initializr使用默认配置后点击Next，如下图所示：



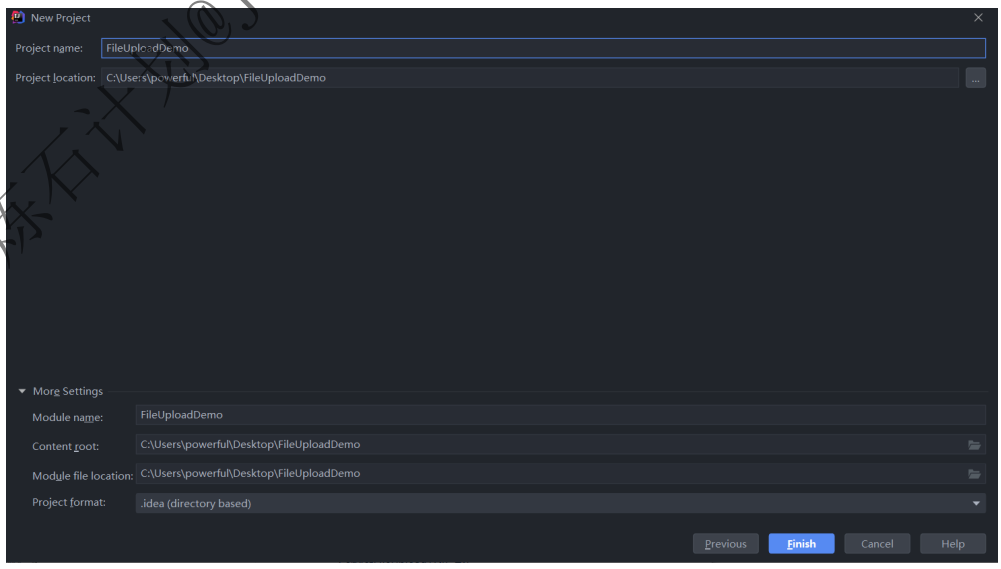
②、下面填写项目信息，修改`Java Version`版本，其他默认即可，如下图所示：



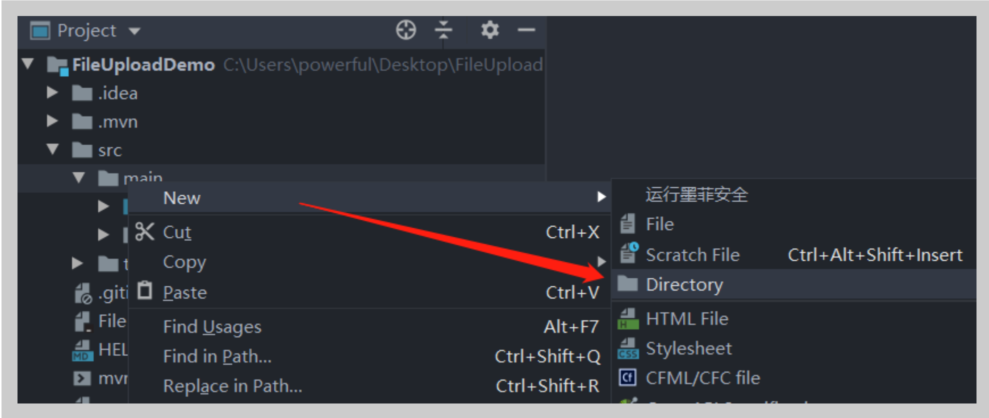
③、依赖选择添加Web -> Spring Web，如下图所示：

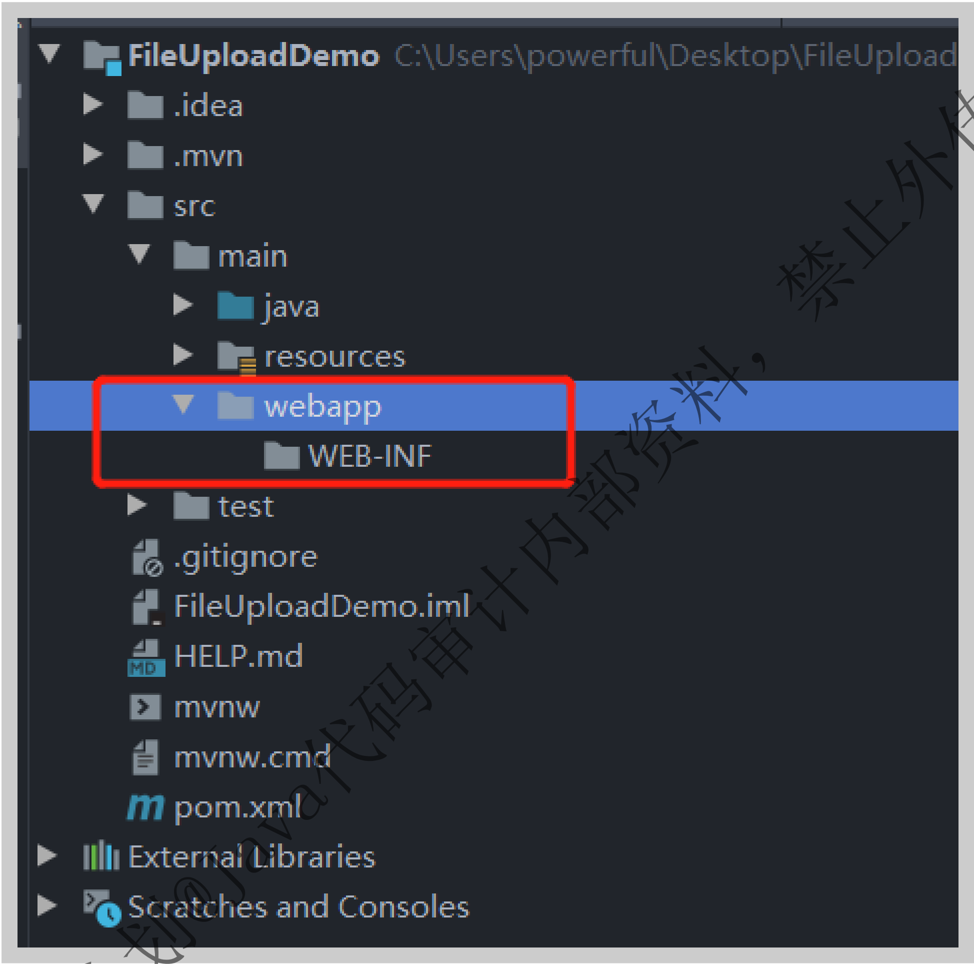


④、下面填写下项目名称和存放地址，点击Finish，即成功初始化创建项目，如下图所示：



⑤、完善下项目结构。在main目录下新建 `webapp` 目录，然后在webapp目录下新建 `WEB-INF` 目录。如下图所示：



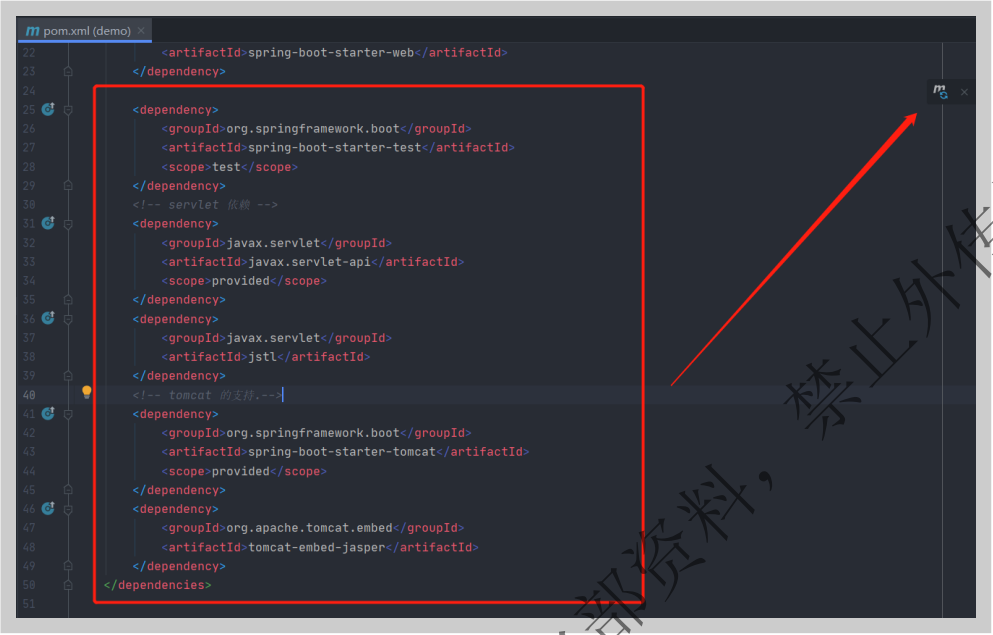


备注： `WEB-INF` 目录为JAVA WEB中安全目录，该目录仅允许服务端访问，客户端无法访问。该目录下有web.xml文件。

⑥、在 `pom.xml` 文件中的 `<dependencies>...</dependencies>` 标签内添加JSP依赖，并重载maven为了下载所添加的依赖。如下图所示：

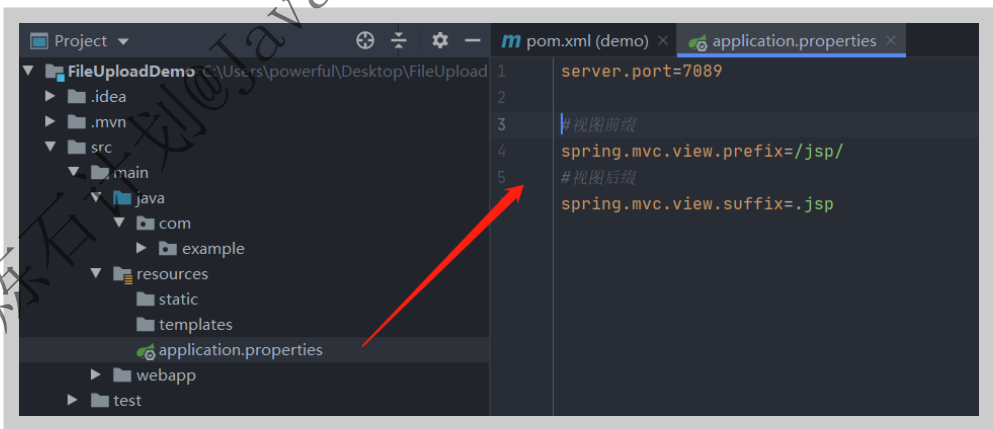
```
<!-- servlet 依赖 -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
<!-- tomcat 的支持.-->
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

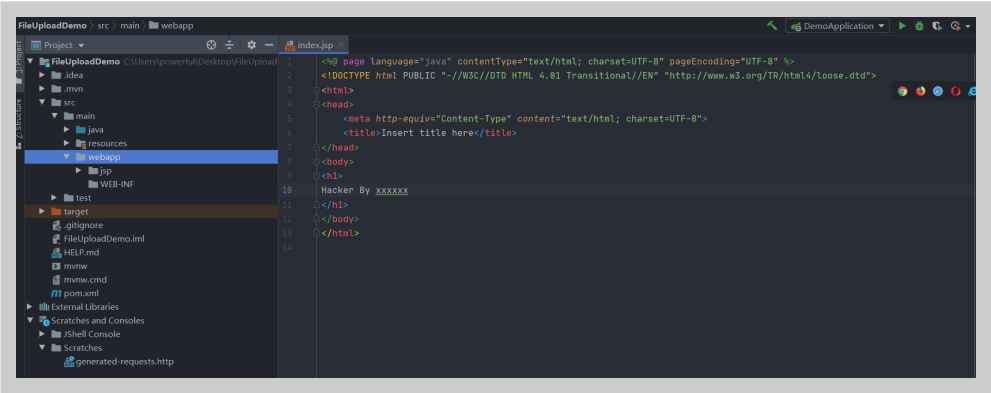


⑦、在 `application.properties` 添加一些配置信息，如下图所示：

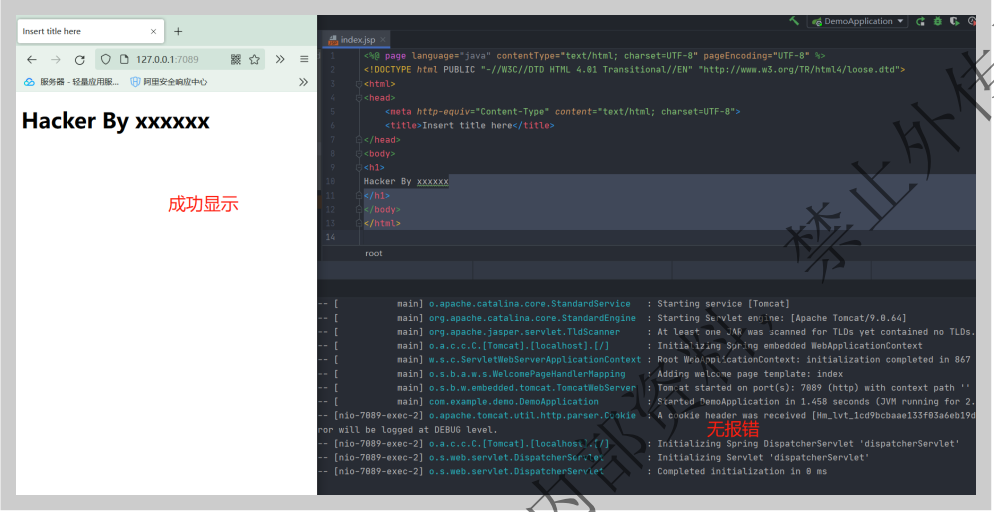
```
server.port=7089
#视图前缀
spring.mvc.view.prefix=/jsp/
#视图后缀
spring.mvc.view.suffix=.jsp
```



⑧、在webapp文件下创建jsp目录，并新建一个index.jsp文件，键入以下信息，如下图所示：



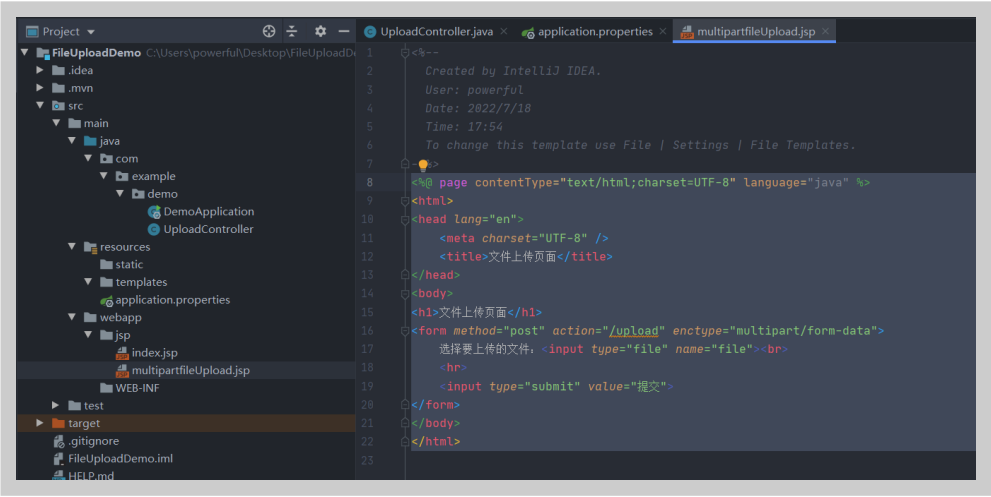
⑨、启动运行，如成功搭建，响应如下图所示：



3、代码示例

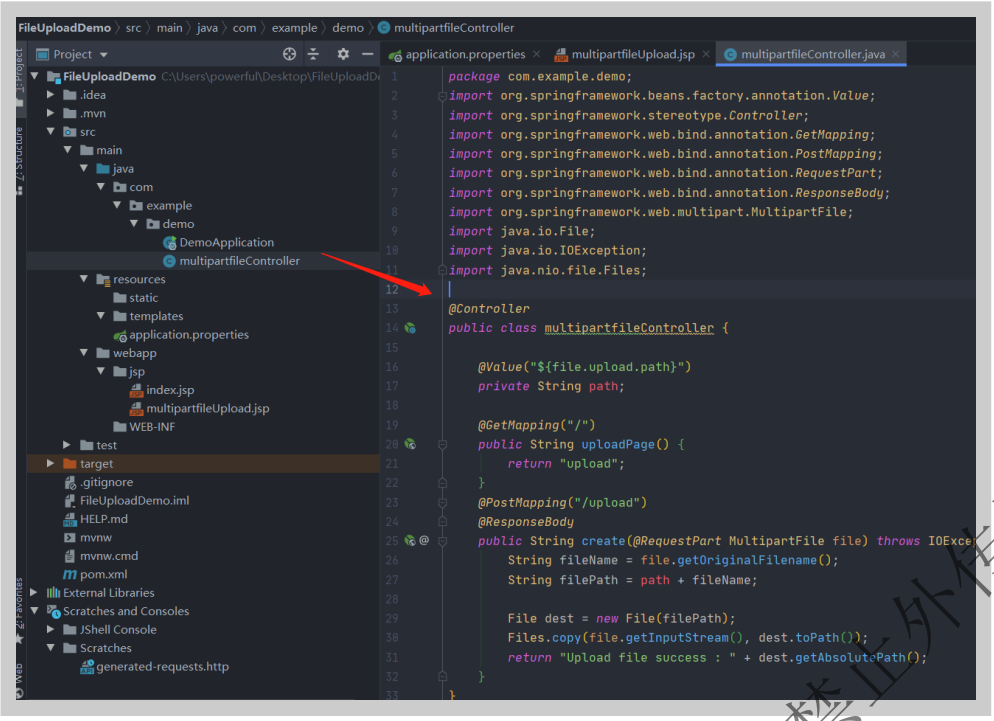
①、在upload文件夹下新建一个名为 `multipartfileUpload.jsp` 的文件，键入以下代码。如下图所示：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head lang="en">
    <meta charset="UTF-8" />
    <title>文件上传页面</title>
</head>
<body>
<h1>文件上传页面</h1>
<form method="post" action="/upload" enctype="multipart/form-data">
    选择要上传的文件：<input type="file" name="file"><br>
    <hr>
    <input type="submit" value="提交">
</form>
</body>
</html>
```

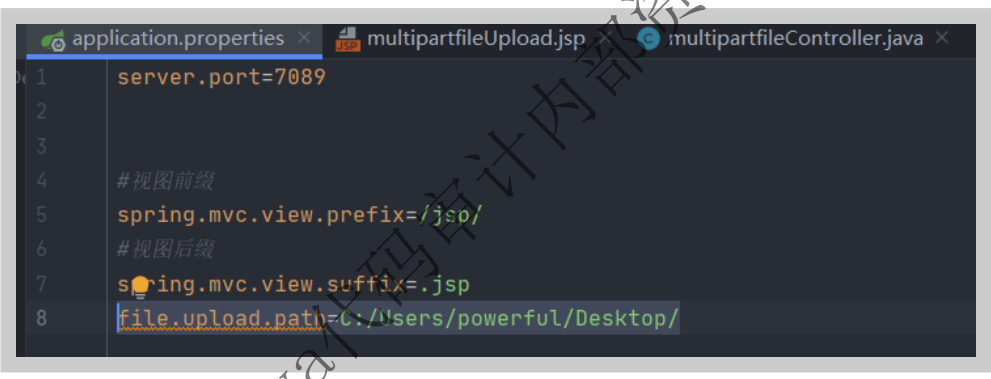


②、在 `com.example.demo` 目录下右键新建一个class名为 `multipartfileController` ，键入以下代码，如下图所示：

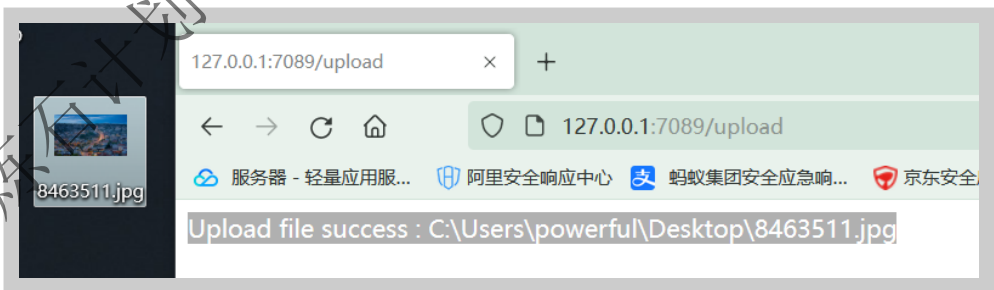
```
package com.example.demo;

import org.springframework.beans.factory.annotation.Value;
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.RequestPart;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;

@Controller
public class multipartfileController {

    @Value("${file.upload.path}")
    private String path;

    @GetMapping("/")
    public String uploadPage() {
        return "upload";
    }

    @PostMapping("/upload")
    @ResponseBody
    public String create(@RequestPart MultipartFile file) throws
IOException {
        String fileName = file.getOriginalFilename();
        String filePath = path + fileName;

        File dest = new File(filePath);
        Files.copy(file.getInputStream(), dest.toPath());
        return "Upload file success : " + dest.getAbsolutePath();
    }

}
```



③、在 `application.properties` 配置文件中添加一条配置信息 `file.upload.path=C:/Users/powerful/Desktop/` ，是上传文件存放的地址，如下图所示：



④、运行启动项目，访问 `127.0.0.1/jsp/multipartfileUpload.jsp` ，上传文件观察结果（是否将上传的图片存放到了桌面处）。



此案例非常简单，使用了multipartfile方式进行文件上传，配合文件流保存上传内容。

二、ServletFileUpload方式文件上传

1、简介

ServletFileUpload方式文件上传依赖commons-fileupload组件。

对于commons-fileupload组件介绍：FileUpload依据规范RFC1867中”基于表单的HTML文件上载”对上传的文件数据进行解析，解析出来的每个项目对应一个FileItem对象。
每个FileItem都有我们可能所需的属性：获取contentType，获取原本的文件名，获取文件大小，获取FiledName(如果是表单域上传)，判断是否在内存中，判断是否属于表单域等。
FileUpload使用FileItemFactory创建新的FileItem。该工厂可以控制每个项目的创建方式。目前提供的工厂实现可以将项目的数据存储临时存储在内存或磁盘上，具体取决于项目的大小（即数据字节，在指定的大小内时，存在内存中，超出范围，存在磁盘上）。

FileUpload又依赖于Commons IO。

官方网站：

```
https://commons.apache.org/proper/commons-fileupload/
```

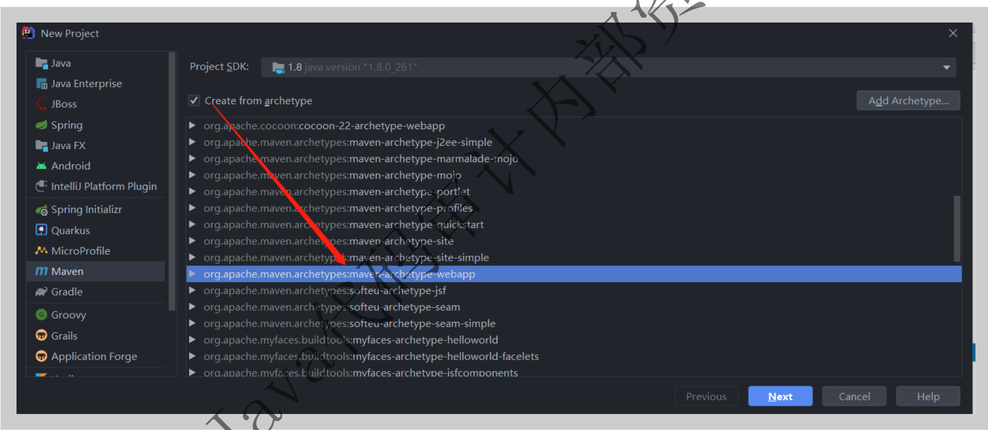
常用的一些方法：

方法	描述
FileItemFactory	表单项工厂接口
ServletFileUpload	文件上传类，用于解析上传的数据
FileItem	表单项类，表示每一个表单项
public List parseRequest(HttpServletRequest request)	解析上传的数据，返回包含 表单项的 List 集合
String FileItem.getFieldName()	获取表单项的 name 属性值
String FileItem.getString()	获取当前表单项的值；
String FileItem.getName()	获取上传的文件名
void FileItem.write(file)	将上传的文件写到 参数 file 所指向存取的硬盘位置

```
FileItemFactory         表单项工厂接口
ServletFileUpload       文件上传类，用于解析上传的数据
FileItem                表单项类，表示每一个表单项
boolean ServletFileUpload.isMultipartContent(HttpServletRequest
request)    判断当前上传的数据格式是否是多段的格式，只有是多段数据，才能使用该方式
public List<FileItem> parseRequest(HttpServletRequest request)
    解析上传的数据，返回包含 表单项的 List 集合
boolean FileItem.isFormField()          判断当前这个表单项，是否是普通的表单
项，还是上传的文件类型，true 表示普通类型的表单项；false 表示上传的文件类型
String FileItem.getFieldName()          获取表单项的 name 属性值
String FileItem.getString()             获取当前表单项的值；

String FileItem.getName()               获取上传的文件名
void FileItem.write( file )             将上传的文件写到 参数 file 所指向存
取的硬盘位置
```

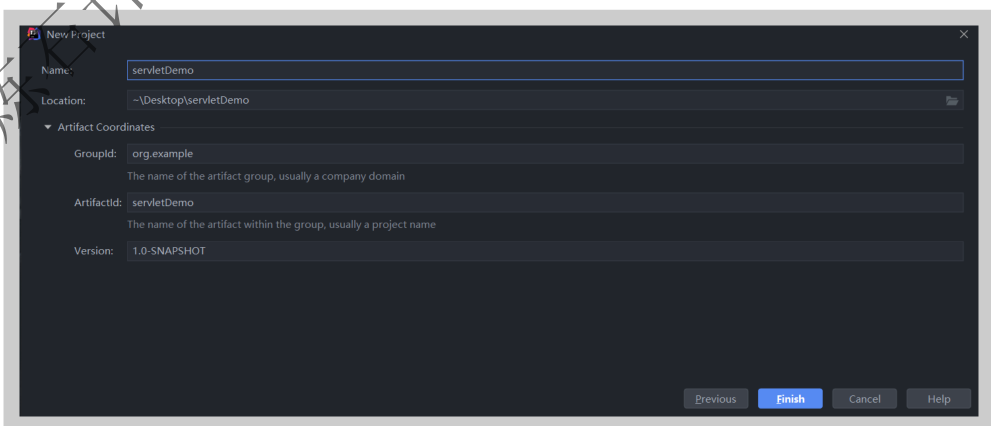
2、创建项目

①、双击启动IDEA，点击 `Create New Project` （如果默认进入某个项目，需要退出，点击左上角 `File -> Close Project` ），左侧选择 `Maven` 项目，选中 `Create from archetype` 后选择 `maven-archetype-webapp` ，点击Next，如下图所示：

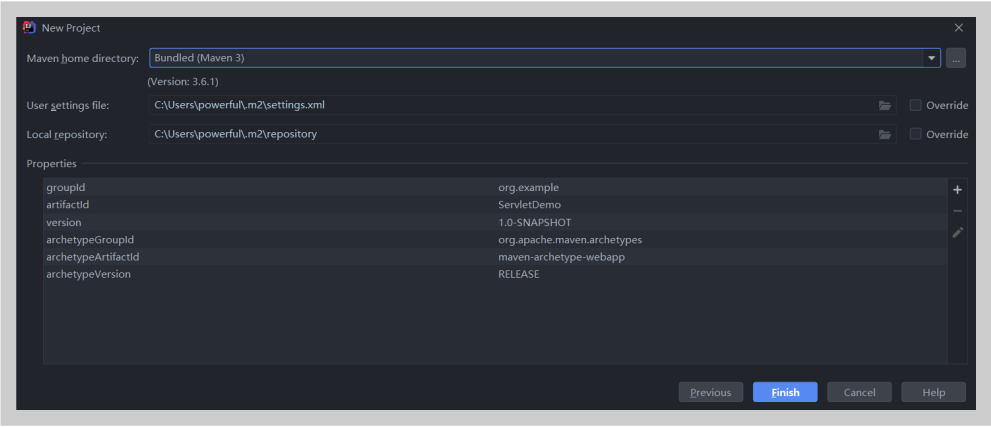


选择 `archetype` 可以简单理解为选择模板，选择不同的模板会有各自的规范。

②、在该页面中起个项目文件名字，其他配置信息默认即可，点击Next，如下图所示：

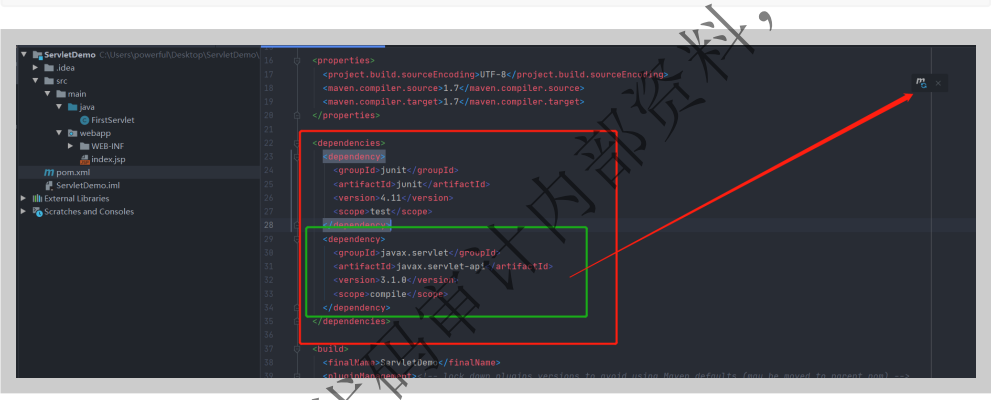


③、这个页面是选择Maven，默认即可，最后点击Finish进入项目，如下图所示：



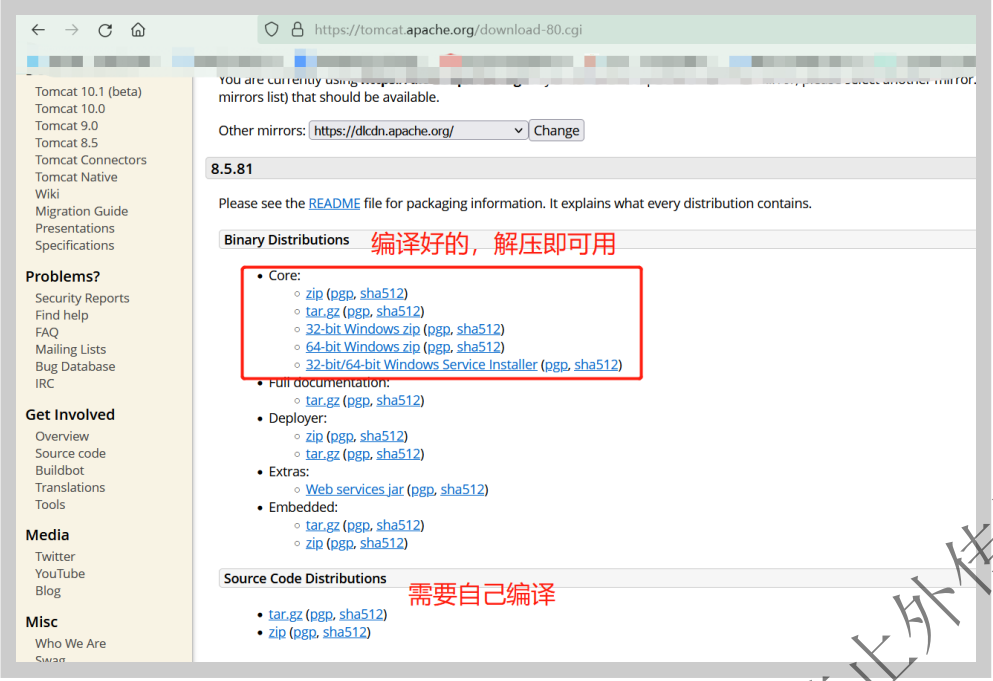
④、在 `pom.xml` 中引入servlet的依赖，写在 `<dependencies>` 标签内，然后可以点击右上角 `加载文件改动` 的按钮（也就是重新加载安装依赖），如下图所示：

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>compile</scope>
</dependency>
```



3、环境配置Tomcat

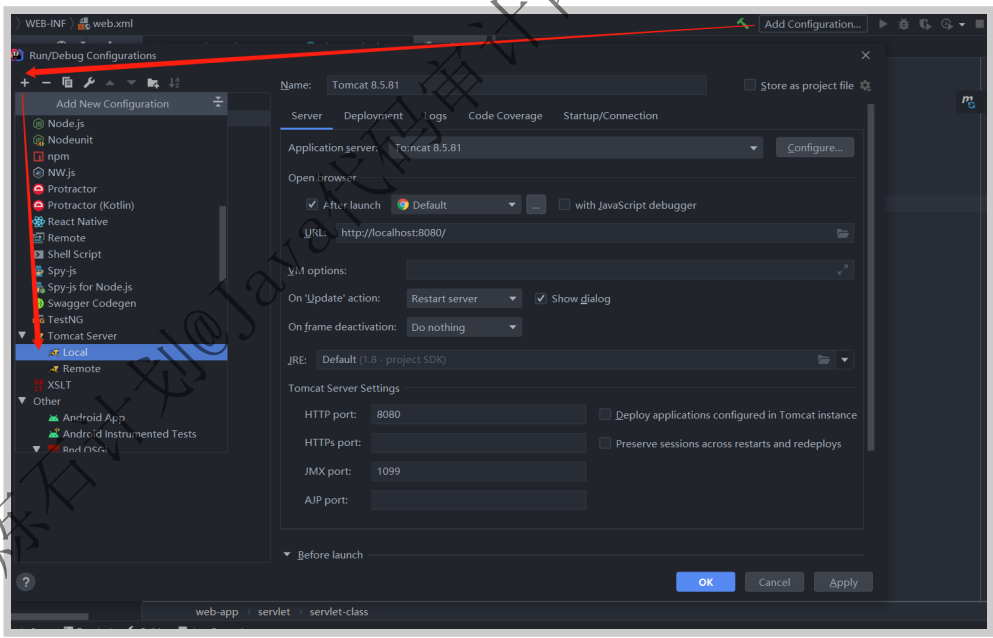
①、访问Tomcat官网 `https://tomcat.apache.org/download-80.cgi` ，下载任意版本Tomcat，我选择的版本是 `8.5.81` ，如下图所示：



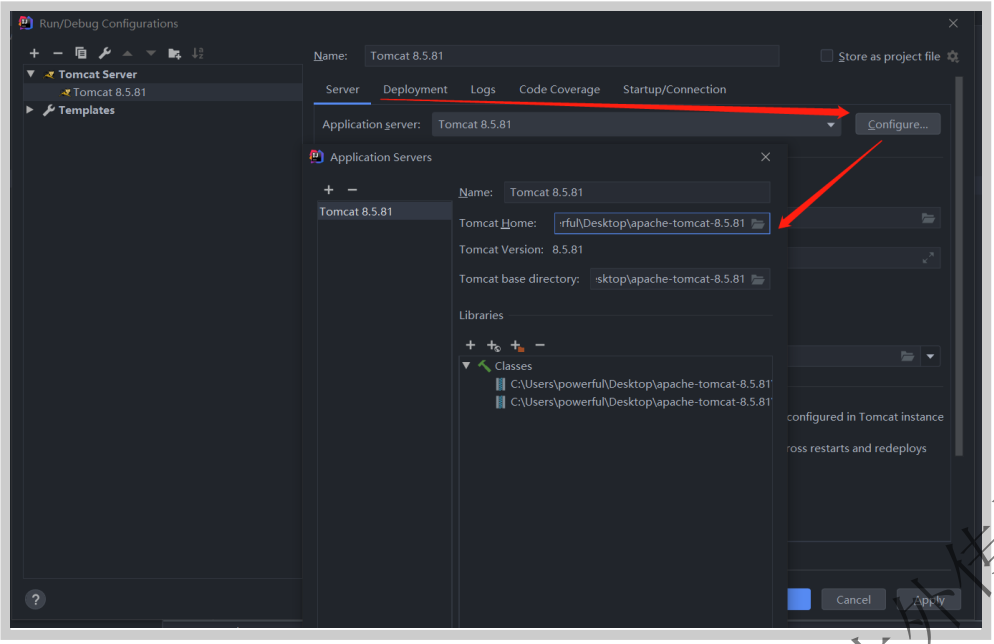
选择 `Binary Distributions` 下的 `Core` 分类，这是Tomcat正式的二进制发布版本，一般学习和开发都用此版本。根据自己计算机系统选择下载项。

②、下载完成后，先解压到桌面备用。

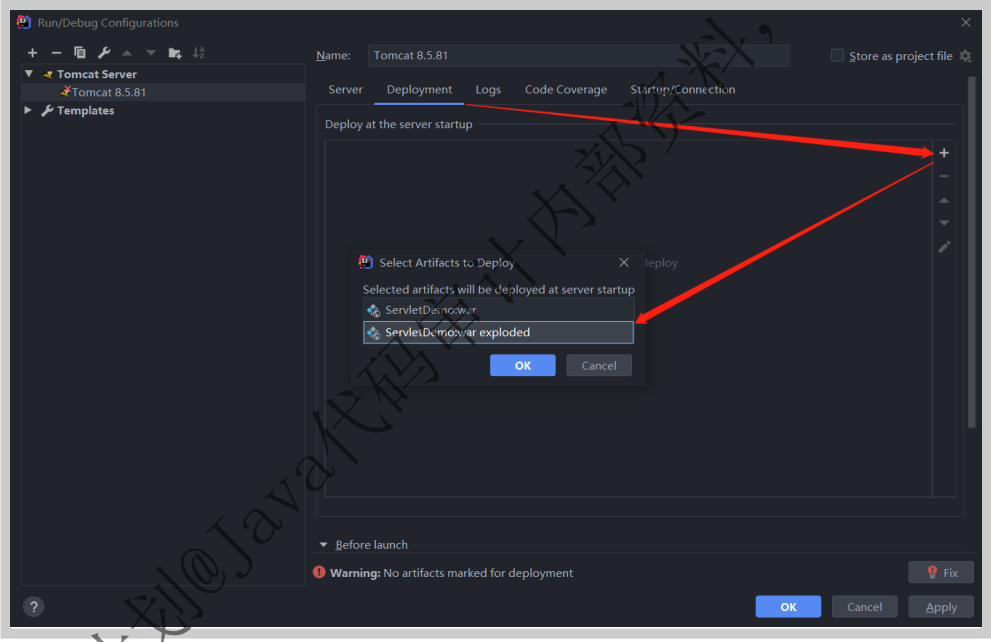
③、进入IDEA，配置Tomcat。在右上侧选择 `Add Configuration...` ，在新的界面点击左上角的 `+` 按钮，滑到下面找到 `Tomcat Server` ，点击选择 `Local` ，如下图所示：



④、在Server标签栏下点击 `Configure...` ，进入新的界面，在 `Tomcat Home` 处添加Tomcat，最后点击OK。如下图所示：

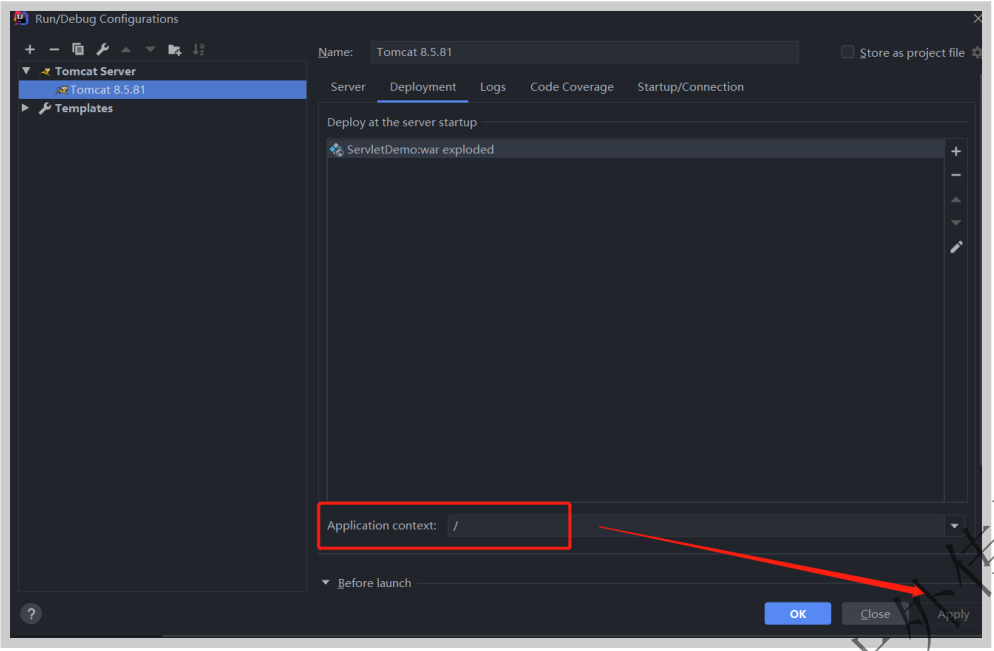


⑤、配置部署方式，选择 `Deployment` 标签栏，在右侧点击 `+` 按钮，选择 `war exploded` 方式，如下图所示：



`war方式`：是发布模式，先打包成war包，再发布。 `war exploded方式` ：常在开发的时候使用这种方式，它可以支持热部署，但需要设置。跟本次案例调试无关，仅是给大家简单讲讲。

⑥、设置URL根路径后，点击 `Apply` ，左后点击 `OK` 。如下图所示：



至此，完成了在IDEA中配置Tomcat。

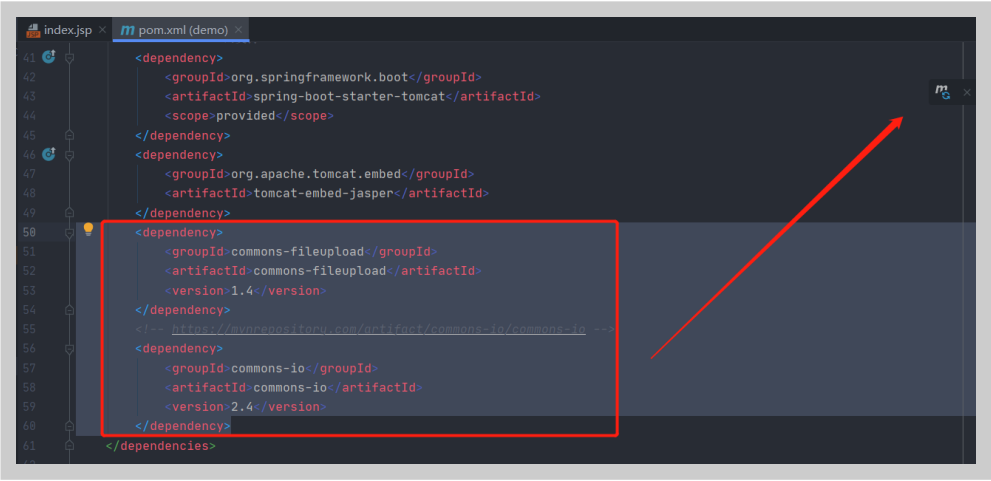
4、代码示例

下文代码示例使用的 `https://www.cnblogs.com/liuyangv/p/8298997.html` 文章中的代码。作者给出了想写的注解，非常适合学习。

配合上面项目搭建，调试一下吧。

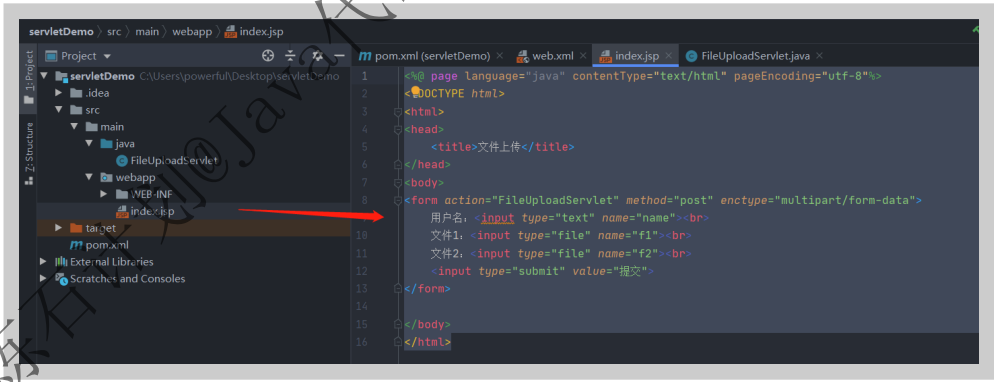
①、首先在pom.xml中添加所需依赖后重新加载maven，如下图所示：

```
        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
            <version>1.4</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/commons-io/commons-io -
->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.4</version>
        </dependency>
```



②、在 `index.jsp` 文件中键入以下代码，如下图所示：

```
<%@ page language="java" contentType="text/html" pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>文件上传</title>
</head>
<body>
<form action="FileUploadServlet" method="post" enctype="multipart/form-data">
    用户名：<input type="text" name="name"><br>
    文件1：<input type="file" name="f1"><br>
    文件2：<input type="file" name="f2"><br>
    <input type="submit" value="提交">
</form>
</body>
</html>
```



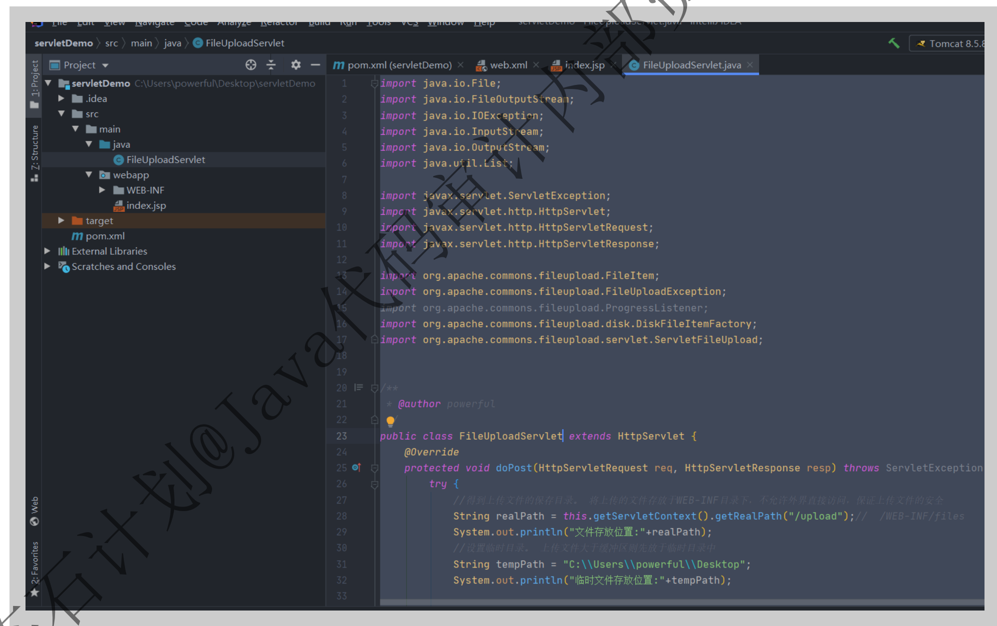
③、右键main目录新建一个名为java的目录，并在该目录下创建一个class，名为 `FileUploadServlet` ，并键入一下代码，如下图所示：

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.ProgressListener;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;


/**
 * @author powerful
 */
public class FileUploadServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        try {
            //得到上传文件的保存目录。 将上传的文件存放于WEB-INF目录下，不允许外
界直接访问，保证上传文件的安全
            String realPath =
this.getServletContext().getRealPath("/upload");//  /WEB-INF/files
            System.out.println("文件存放位置:"+realPath);
            //设置临时目录。 上传文件大于缓冲区则先放于临时目录中
            String tempPath = "C:\\Users\\powerful\\Desktop";
            System.out.println("临时文件存放位置:"+tempPath);


            //判断存放上传文件的目录是否存在（不存在则创建）
            File f = new File(realPath);
            if(!f.exists()&&!f.isDirectory()){
                System.out.println("目录或文件不存在! 创建目标目录。");
                f.mkdir();
            }
            //判断临时目录是否存在（不存在则创建）
            File f1 = new File(tempPath);
            if(!f1.isDirectory()){
                System.out.println("临时文件目录不存在! 创建临时文件目录");
                f1.mkdir();


            /**
             * 使用Apache文件上传组件处理文件上传步骤:
             *
             * */
            //1、设置环境:创建一个DiskFileItemFactory工厂
            DiskFileItemFactory factory = new DiskFileItemFactory();

            //设置上传文件的临时目录
            factory.setRepository(f1);

            //2、核心操作类:创建一个文件上传解析器。
            ServletFileUpload upload = new ServletFileUpload(factory);
            //解决上传"文件名"的中文乱码
            upload.setHeaderEncoding("UTF-8");

            //3、判断enctype:判断提交上来的数据是否是上传表单的数据
            if(!ServletFileUpload.isMultipartContent(req)){
                System.out.println("不是上传文件，终止");
                //按照传统方式获取数据
```

```
                return;
            }

            //==获取输入项==
//              //限制单个上传文件大小(5M)
//              upload.setFileSizeMax(1024*1024*4);
//              //限制总上传文件大小(10M)
//              upload.setSizeMax(1024*1024*6);

            //4、使用ServletFileUpload解析器解析上传数据，解析结果返回的是一个
List<FileItem>集合，每一个FileItem对应一个Form表单的输入项
            List<FileItem> items =upload.parseRequest(req);
            for(FileItem item:items){
                //如果fileitem中封装的是普通输入项的数据（输出名、值）
                if(item.isFormField()){
                    String filedName = item.getFieldName();//普通输入项数
据的名
                    //解决普通输入项的数据的中文乱码问题
                    String filedValue = item.getString("UTF-8");//普通输
入项的值
                    System.out.println("普通字
段:"+filedName+"=="+filedValue);
                }else{
                    //如果fileitem中封装的是上传文件，得到上传的文件名称，
                    String fileName = item.getName();//上传文件的名
                    //多个文件上传输入框有空 的异常处理
                    if(fileName==null||"".equals(fileName.trim())){  //
去空格是否为空
                        continue;//为空，跳过当次循环，   第一个没输入则跳过可
以继续输入第二个
                    }

                    //注意：不同的浏览器提交的文件名是不一样的，有些浏览器提交上
来的文件名是带有路径的，如：  c:\a\b\1.txt，而有些只是单纯的文件名，如：1.txt
                    //处理上传文件的文件名的路径，截取字符串只保留文件名部分。//
截取留最后一个"\"之后，+1截取向右移一位（"\a.txt"-->"a.txt"）
                    fileName =
fileName.substring(fileName.lastIndexOf("\\")+1);
                    //拼接上传路径。存放路径+上传的文件名
                    String filePath = realPath+"\\"+fileName;
                    //构建输入输出流
                    InputStream in = item.getInputStream(); //获取item中
的上传文件的输入流
                    OutputStream out = new FileOutputStream(filePath);
//创建一个文件输出流

                    //创建一个缓冲区
                    byte b[] = new byte[1024];
                    //判断输入流中的数据是否已经读完的标识
                    int len = -1;
                    //循环将输入流读入到缓冲区当中，(len=in.read(buffer))!
=-1就表示in里面还有数据
                    while((len=in.read(b))!=-1){  //没数据了返回-1
                        //使用FileOutputStream输出流将缓冲区的数据写入到指定
的目录(savePath+"\\"+filename)当中
                        out.write(b, 0, len);
                    }
                    //关闭流
```

```
                        out.close();
                        in.close();
                        //删除临时文件
                        try {
                            Thread.sleep(3000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        item.delete();//删除处理文件上传时生成的临时文件
                        System.out.println("文件上传成功");
                    }
                }
            } catch (FileUploadException e) {
                //e.printStackTrace();
                throw new RuntimeException("服务器繁忙，文件上传失败");
            }
        }

        @Override
        protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
            this.doPost(req, resp);
        }
}
```



④、运行项目，上传文件观察结果。

至此，两种上传方式均已演示完毕。大家调试下代码学习一下吧。

三、拓展文件上传项目学习

1、SpringBoot集成上传文件，该项目多是使用的Multipartfile方式进行的文件上传。但其中有一块是使用了文件流方式，位于`/src/main/java/com/example/springbootupload/controller/FileUploadController.java`文件内，大家可以调试一下。

```
https://github.com/xiaonongOne/springboot-upload
```

2、MyUploader-Backend项目实现了单文件上传，多文件上传，大文件上传，断点续传，文件秒传，图片上传。完整项目建议学习。

```
https://github.com/gaoyuyue/MyUploader-Backend
```