## 文件上传

### 需要导入jar包

commons-fileupload-1.3.1.jar和commons-io-2.4.jar

### 基于表单的文件上传

文件上传的表单例子如下：

<form action="upload" method="post" enctype="multipart/form-data">

<input type="file" name="myfile"/>

……

</form>

**注意：**enctype属性指定的是**表单数据的编码方式**，multipart/form-data这种编码方式会**以二进制流的方式处理表单数据**

### MultipartFile接口

1）在Spring MVC框架中，**上传文件时**，**将文件相关信息及操作封装到MultipartFile对象中**。因此，开发者只需要使用MultipartFile类型声明模型类的一个属性，即可以对被上传文件进行操作。

2）上传文件时，需要在配置文件中使用spring的CommosMultipartResolver 配置MultipartResolver 用于文件上传。

**注意：**该接口有以下方法：

String getOriginFilename():返回客户端“原始上传的”文件名字

Long getSize():返回这个文件的大小，单位字节

Boolean isEmpty():判断上传文件是否为空

Void transferTo():将上传文件保存到指定目录下

### 1.4 request.getServletPath()和request.getContextPath()

若请求[http://localhost:8080/testweb/abc](http://localhost:8080/testweb/abc" \t "https://blog.csdn.net/qq_27770257/article/details/_blank)，则依次为：

request.getContextPath()：/testweb————项目名

request.getRequestURI()：/testweb/abc

request.getRequestURL()：[http://localhost:8080/testweb/abc](http://localhost:8080/testweb/abc" \t "https://blog.csdn.net/qq_27770257/article/details/_blank)

request.getServletContext().getRealPath(“/”)：机器中的真实路径G:\java\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\testweb\

**比如：在工程的webapp下新建一个upload的目录**

String imgName=pic1.getOriginalFilename();

String filepath=request.getServletContext().getRealPath("/")+"upload"+"\\"+imgName;

File file=new File(filepath);

pic1.transferTo(file);

1. 文件下载
2. 遇到的错误

3.1 配置文件报错





### 1.5 get乱码

对于get请求中文参数出现乱码解决方法**有两个**：

（1）：修改tomcat配置文件添加编码与工程编码一致，如下：

<Connector URIEncoding="utf-8" connectionTimeout="20000" port="8080"protocol="HTTP/1.1" redirectPort="8443"/>

（2）另一种方法就是对参数进行重新编码：因为我们都知道，tomcat一般情况下代码都是ISO8859-1编码

所以我们要对参数重新进行编码

String userName = new String(request.getParamter("userName").getBytes("ISO8859-1"),"utf-8");

ISO8859-1是tomcat默认编码，需要将tomcat编码后的内容按utf-8编码

### 1.6 Post乱码

解决post请求的中文乱码问题

在项目中web.xml配置文件中加入编码过滤器即可：

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

以上可以解决post请求乱码问题

## 2、 [彻底解决springMVC中文乱码](https://www.cnblogs.com/javahr/p/8269669.html)

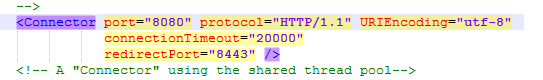
### 第一步：jsp页面编码

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>

### 第二步：解决url的乱码（get请求）

改tomcat中server.xml中Connector的port=“8080”，加上一个 URIEncoding=”utf-8”



### 第三步：配置过滤器，指定所有请求的编码

（1）配置spring的编码过滤器，为了防止spring中post方式提交的时候中文乱码，方法：修改web.xml文件，添加spring的编码过滤器

<!-- 配置编码方式过滤器,注意一点:要配置在所有过滤器的前面 -->

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

（2）配置编码过滤器，方法：先创建filter类，再修改web.xml文件，注意的是放在spring的编码过滤器之后

filter类：

public class EncoidingFilter implements Filter {

private String encoding="utf-8";

@Override

public void destroy() {

// TODO Auto-generated method stub

}

//过滤方法 是否往下执行

@Override

public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain chain)

throws IOException, ServletException {

HttpServletRequest request=(HttpServletRequest)arg0;

HttpServletResponse response=(HttpServletResponse)arg1;

request.setCharacterEncoding(encoding);

response.setCharacterEncoding(encoding);

//过滤通行证

chain.doFilter(request, response);

}

//根据web.xml文件的配置进行初始化

@Override

public void init(FilterConfig arg0) throws ServletException {

this.encoding = arg0.getInitParameter("Encoding");

}

}

**修改web.xml，添加如下配置，注意的是放在spring的编码过滤器之后：**

<!-- 配置编码过滤 -->

<filter>

<filter-name>EncoidingFilter</filter-name>

<filter-class>com.qiyuan.filter.EncoidingFilter</filter-class>

<init-param>

<param-name>Encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>EncoidingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

### 第四步：文件编码

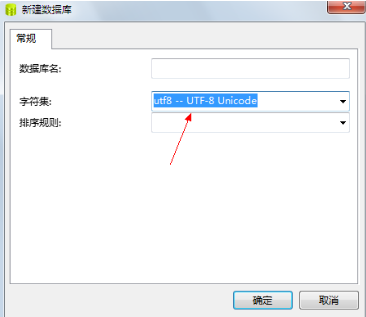
将文件另存为utf-8格式

### 第五步：数据库编码

**1）JDBC连接的时候**：指定编码格式

public static String URL="jdbc:mysql://127.0.0.1:3306/mvcdb? useUnicode=true &**characterEncoding=UTF-8**"

**2）创建数据库的时候指定utf-8编码格式**



### 第六步：Eclipse修改编码格式

**1）修改工作空间默认编码**

在菜单导航栏上Window-->Preferences 打开"首选项"对话框，左侧导航树，导航到 General-->Workspace，改为utf-8

**2）修改项目的编码**

在Eclipse项目文件上右键,选择Properties,在Resource选择修改编码格式

**3）修改某文件类型的编码**

有时候我们需要某种类型的文件,如:\*.jsp、\*.java等

导航栏window-->preferences打开"首选项"对话框，左侧导航树，导航到 Genera-->Content Types

我这边以\*.java为例，在下面的Default encoding，输入框中输入UTF-8->Update->OK

**4）修改单个文件的编码**

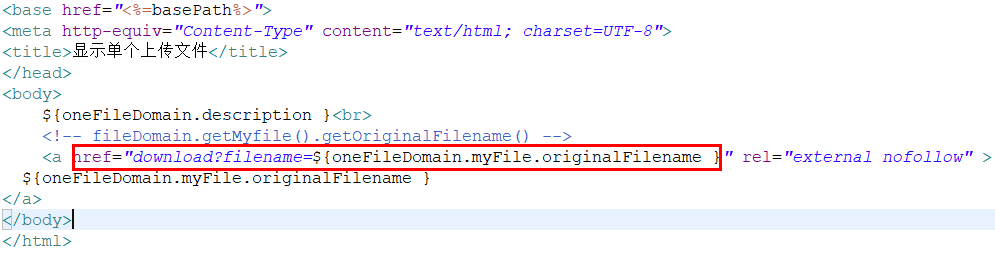
在包资源管理器右键点击文件->属性，改变文本文件编码格式为UTF-8

经过上述步骤，新建java文件即为UTF-8编码，新建项目也是UTF-8编码,在

Eclipse编译、运行、调试都是没问题的。

## 3、文件下载

### 下载页面的jsp，点击文件下载



### 处理下载请求的/download的方法

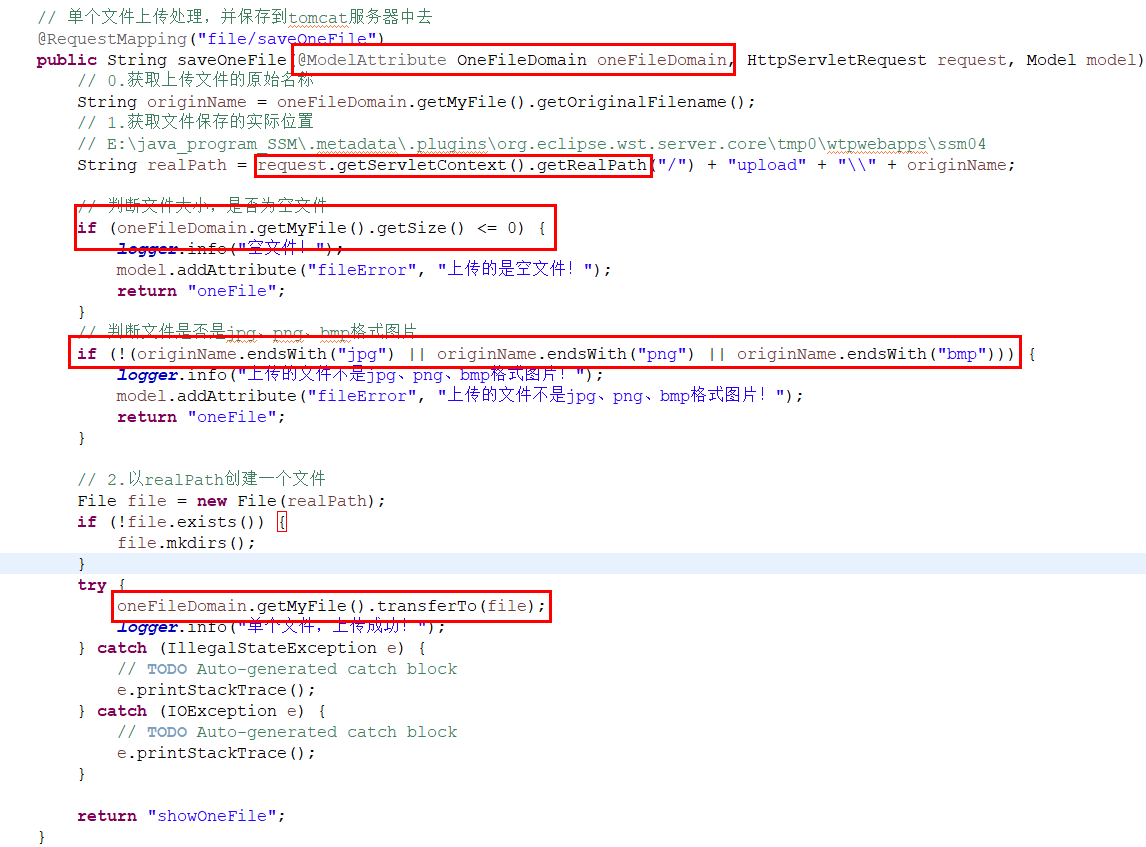
|  |
| --- |
| // 文件的下载  @RequestMapping(value = "/download")  **public** **void** downFile(HttpServletRequest request, HttpServletResponse response) **throws** Exception {  //浏览器中传递过来的fileName编码是"ISO8859-1"，需要解码之后，在使用utf-8编码  String fileName = request.getParameter("filename");  fileName = **new** String(fileName.getBytes("ISO8859-1"), "utf-8");  String realPath = request.getSession().getServletContext().getRealPath("/upload");  String filePath = realPath + File.***separator*** + fileName;    //使用二进制流方式读取文件BufferedInputStream  InputStream fis = **null**;  **byte**[] buffer = **null**;  **try** {    //将文件内容以字节流方式读到buffer字节数组中去  fis = **new** BufferedInputStream(**new** FileInputStream(filePath));  buffer = **new** **byte**[fis.available()];  fis.read(buffer);    } **finally** {  **try** {  **if** (fis != **null**) {  fis.close();  }  } **catch** (Exception e) {  }  }    response.setContentType("application/txt");  response.addHeader("Content-Disposition","attachment;filename=" + **new** String(fileName.getBytes(), "ISO8859-1"));  response.addHeader("Content-Length", "" + buffer.length);    //使用二进制流方式将文件输出到response.getOutputStream()  OutputStream os = **new** BufferedOutputStream(response.getOutputStream());  os.write(buffer, 0, buffer.length);  os.flush();  os.close();  } |

## 实战项目（部分截图）

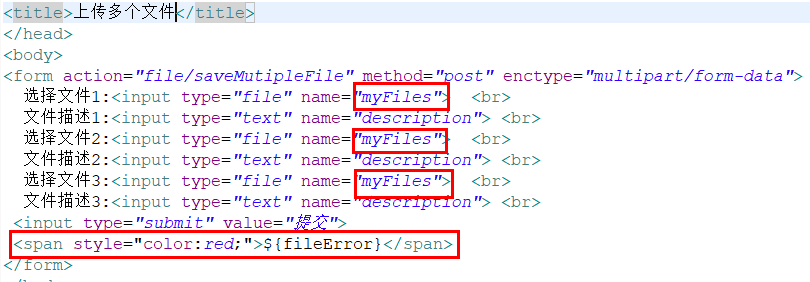
**单文件上传：**



**单文件上传并处理的方法：**



**多文件上传的jsp页面：**



**多文件上传处理controller方法：**



**显示多个文件：**

