# 尚马教育 JAVA WEB课程

# 文件上传、过滤器、监听器

文档编号：B07

创建日期： 2017-06-13

最后修改日期：2019-09-23

版 本 号：V2.0

电子版文件名：尚马教育-第二阶段-8.文件上传、过滤器、监听器.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2018-08-01 | 王绍成 | 初始版本 | V1.0 |
| 2019-09-13 | 郭雪岩 | 内容更新 | V2.0 |

**主讲人：**

**郭雪岩**

目录

[1. 文件上传](#_Toc27905_WPSOffice_Level1) [3](#_Toc27905_WPSOffice_Level1)

[1.1. 介绍](#_Toc18199_WPSOffice_Level2) [3](#_Toc18199_WPSOffice_Level2)

[1.2. 使用工具](#_Toc27619_WPSOffice_Level2) [3](#_Toc27619_WPSOffice_Level2)

[1.3. 文件上传](#_Toc16189_WPSOffice_Level2) [3](#_Toc16189_WPSOffice_Level2)

[1.4. Ajax文件上传](#_Toc3298_WPSOffice_Level2) [4](#_Toc3298_WPSOffice_Level2)

[2. Filter过滤器](#_Toc18199_WPSOffice_Level1) [5](#_Toc18199_WPSOffice_Level1)

[2.1. 什么是过滤器](#_Toc26117_WPSOffice_Level2) [5](#_Toc26117_WPSOffice_Level2)

[2.2. 过滤器生命周期](#_Toc2262_WPSOffice_Level2) [5](#_Toc2262_WPSOffice_Level2)

[2.3. 过滤器配置](#_Toc30823_WPSOffice_Level2) [5](#_Toc30823_WPSOffice_Level2)

[2.4. 过滤器典型案例](#_Toc14740_WPSOffice_Level2) [6](#_Toc14740_WPSOffice_Level2)

[3. 监听器listener](#_Toc27619_WPSOffice_Level1) [7](#_Toc27619_WPSOffice_Level1)

[3.1. 什么是监听器](#_Toc16028_WPSOffice_Level2) [7](#_Toc16028_WPSOffice_Level2)

[3.2. 监听器配置和使用](#_Toc11592_WPSOffice_Level2) [7](#_Toc11592_WPSOffice_Level2)

## 文件上传

### 介绍

* 在很多业务场景,用户都需要上传头像、证件照以及其他文件.
* 文件上传就是用户通过浏览器选择需要上传的文件并将文件传输至网站服务器上的一个过程

### 使用工具

文件上传需要对请求中传的的文件流进行解析，一般会选择工具来对文件流进行解析

* 文件上传相关的jar包

commons-fileupload-1.3.3.jar

commons-io-2.5.jar

### 文件上传

* 文件上传请求由浏览器发起

|  |
| --- |
| <form action="uploadServlet" method="post" enctype="multipart/form-data">  <input type="text" name="uname"><br>  <input type="file" name="uimg"><br>  <input type="submit">  </form> |

需要文件选择框并且设置了name属性，form的method为post，加设置enctype="multipart/form-data"，此时form提交时，会把选择到的文件以二进制数据形式发送到服务器，否则不会发送文件

* 服务器需要对请求中的文件数据进行解析，还原成文件，这个由jar包中提供的类完成

|  |
| --- |
| String basePath = getServletContext().getRealPath("/");//获取实际路径（本地写文件）  String myFolder = "upload"; //统一存放上传文件的目录（自定义）  //DiskFileItemFactory 临时文件工厂类  DiskFileItemFactory dff = new DiskFileItemFactory();  //上传文件核心对象  ServletFileUpload sfu = new ServletFileUpload(dff);  try {  //把流中的数据 按照临时文件工厂格式 格式化成文件对象  List<FileItem> lf = sfu.parseRequest(req);  for(FileItem fi:lf) {  //如果是文件 FormField的值是false 如果是表单传递的普通数据 FormField的值是true  if(!fi.isFormField()) {//如果是文件对象，把临时文件内容写入本地文件  File myfile = new File(basePath+myFolder+"/"+fi.getName());  fi.write(myfile);  }else {//如果是普通数据 可以通过下边两个方法取值  fi.getFieldName();//传递的key  fi.getString();//传递的value  }  }  } catch (FileUploadException e) {  e.printStackTrace();  } catch (Exception e) {  e.printStackTrace();  } |

### Ajax文件上传

* 文件上传使用form传递，普通数据取值变得非常麻烦，可以把文件上传和普通表单提交的流程分开，通过工具把文件单独上传到服务器（异步上传），并在页面接收存储的地址（相对路径），用户填完数据提交表单时，文件已上传过，只需要提交普通表单即可。

|  |
| --- |
| 通过工具ajaxFileUpload 上传文件  $.ajaxFileUpload({  url:"ajaxUpload",//地址  type:"post",//请求类型  fileElementId:"uimgfile", //指定文件对象  secureuri:false,  data:{}, //请求的参数 键值对字符串 json格式  dataType:"text",//ajaxFileUpload 需要自己解析json 设置成text 返回数据的类型  success:function(data){//响应成功时的回调函数  console.log(data);  var jsonData = JSON.parse(data);  //将返回的数据填入表单隐藏域，并生成预览（根据路径显示图片）  $("#uimg").val(jsonData.imgsrc);  $("#showUploadImg").html("<img src='"+jsonData.imgsrc+"'/>")  }  }); |

## Filter过滤器

### 什么是过滤器

* 过滤器是向 Web 应用程序的请求和响应处理添加功能的Web 服务组件
* Filter是一个可以加在servlet前执行的组件，可以通过这个机制制作一些通用性的功能，跟生活中用的水的过滤器的机制很相似。

### 过滤器生命周期

|  |
| --- |
| **实例化**  **初始化**  **过 滤**  **销 毁**  **init ( …)**  **doFilter (… )**  **destory ( )**  **Filter 接口** |

Filter跟servlet生命周期类似

### 过滤器配置

* 过滤器需要实现filter接口，dofilter方法中写具体要实现的功能

|  |
| --- |
| public class MyFilter implements Filter  public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2) |

注意：过滤器只提供机制，功能需要自己编写

* Web.xml中配置，配置格式与servlet相似

|  |
| --- |
| <filter>  <filter-name>myFilter2</filter-name>  <filter-class>com.javasm.filter.MyFilter2</filter-class>  </filter>  <filter-mapping>  <filter-name>myFilter2</filter-name>  <url-pattern>/testServlet</url-pattern>  </filter-mapping> |

注意：过滤器的url-pattern为了制作通用功能，通常会配置为<url-pattern>/\*</url-pattern>

如果设置为/\*所有访问服务器的请求都会先进入此过滤器，包括请求静态资源（html、css、js、图片、等等）

* 过滤器允许配置多个，执行顺序由配置的先后顺序决定

### 过滤器典型案例

* 过滤器这种机制比较适合做通用功能

统一编码设置

权限控制

## listener监听器

### 什么是监听器

监听器是javaweb组件中提供的另外一种机制，用来监控域对象的创建、销毁过程（在发生创建、销毁时，可以执行自己的代码）以及域对象中属性的创建、替换、销毁。

* 监听web对象创建与销毁的监听器

        ServletContextListener

        HttpSessionListener

        ServletRequestListener

* 监听web对象属性变化

ServletContextAttributeListener

HttpSessionAttributeListener

ServletRequestAttributeListener

### 监听器配置和使用

HttpSessionListener为例

|  |
| --- |
| public class MyListener implements HttpSessionListener {  @Override  public void sessionCreated(HttpSessionEvent arg0) {  //session创建时执行的代码  }  @Override  public void sessionDestroyed(HttpSessionEvent arg0) {  //session销毁时执行的代码  }  } |

Web.Xml中配置

|  |
| --- |
| <listener>  <listener-class>com.javasm.listener.MyListener</listener-class>  </listener> |