回顾：

1. 监听器
2. 生命周期监听器

ServletRequestListener

HttpSessionListener

ServletContextListener

1. 属性监听器

ServletRequestAttributeListener

HttpSessionAttributeListener

ServletContextAttributeListener

1. Session相关监听器
2. 案例： 显示在线用户!

监听器：ServletContextListener/HttpSessionListener

1. 国际化
2. 国际化资源文件： 基础名\_语言\_国家.properties 【通用】

基础名.properties (默认)

1. 获取国际化文本
   1. ResourceBundle工具类
2. JSP页面国际化
   1. Jstl提供的“国际化与格式化标签库”

# 1. 文件上传与下载

## 1.1 文件上传

案例：

注册表单/保存商品等相关模块！

--🡪 注册选择头像 / 商品图片

(数据库：存储图片路径 / 图片保存到服务器中指定的目录)

### 文件上传，要点:

前台：

1. 提交方式：post

2. 表单中有文件上传的表单项： <input type=”file” />

3. 指定表单类型:

默认类型：enctype="application/x-www-form-urlencoded"

**文件上传类型：multipart/form-data**

### 手动实现文件上传

|  |
| --- |
| <body>  <form name=*"frm\_test"* action=*"*${pageContext.request.contextPath }*/upload"* method=*"post"* enctype=*"multipart/form-data"*>  用户名：<input type=*"text"* name=*"userName"*> <br/>  文件： <input type=*"file"* name=*"file\_img"*> <br/>    <input type=*"submit"* value=*"注册"*>  </form>  </body> |

|  |
| --- |
|  |
| **public** **class** UploadServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  /\*  request.getParameter(""); // GET/POST  request.getQueryString(); // 获取GET提交的数据  request.getInputStream(); // 获取post提交的数据 \*/    /\*\*\*\*\*\*\*\*\*\*\*手动获取文件上传表单数据\*\*\*\*\*\*\*\*\*\*\*\*/    //1. 获取表单数据流  InputStream in = request.getInputStream();  //2. 转换流  InputStreamReader inStream = **new** InputStreamReader(in, "UTF-8");  //3. 缓冲流  BufferedReader reader = **new** BufferedReader(inStream);  // 输出数据  String str = **null**;  **while** ((str = reader.readLine()) != **null**) {  System.*out*.println(str);  }    // 关闭  reader.close();  inStream.close();  in.close();  } |
| 输出结果： |
| ------WebKitFormBoundaryGoQviatB7iM1dhPr  Content-Disposition: form-data; name="userName" 【FileItem】  Jack  ------WebKitFormBoundaryGoQviatB7iM1dhPr  Content-Disposition: form-data; name="file\_img"; filename="reamde.txt"  Content-Type: text/plain 【FileItem】  test!!!!!!!!!!!!!  test!!!!!!!!!!!!!  ------WebKitFormBoundaryGoQviatB7iM1dhPr-- |
| 总结：  最终获取数据，要对上面的结果进行解析！  文件上传，在开发中经常用，每次都写解析程序！(工具类)  也可以使用开源的文件上传组件-FileUpload组件！ |
|  |

### Apache提供的文件上传组件:FileUpload组件

文件上传功能开发中比较常用，apache也提供了文件上传组件！

FileUpload组件:

1. 下载源码

2. 项目中引入jar文件

commons-fileupload-1.2.1.jar 【文件上传组件核心jar包】

commons-io-1.4.jar 【封装了对文件处理的相关工具类】

|  |
| --- |
| **public** **class** UploadServlet **extends** HttpServlet {  // upload目录，保存上传的资源  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    /\*\*\*\*\*\*\*\*\*文件上传组件： 处理文件上传\*\*\*\*\*\*\*\*\*\*\*\*/      **try** {  // 1. 文件上传工厂  FileItemFactory factory = **new** DiskFileItemFactory();  // 2. 创建文件上传核心工具类  ServletFileUpload upload = **new** ServletFileUpload(factory);    // 一、设置单个文件允许的最大的大小： 30M  upload.setFileSizeMax(30\*1024\*1024);  // 二、设置文件上传表单允许的总大小: 80M  upload.setSizeMax(80\*1024\*1024);  // 三、 设置上传表单文件名的编码  // 相当于：request.setCharacterEncoding("UTF-8");  upload.setHeaderEncoding("UTF-8");      // 3. 判断： 当前表单是否为文件上传表单  **if** (upload.*isMultipartContent*(request)){  // 4. 把请求数据转换为一个个FileItem对象，再用集合封装  List<FileItem> list = upload.parseRequest(request);  // 遍历： 得到每一个上传的数据  **for** (FileItem item: list){  // 判断：普通文本数据  **if** (item.isFormField()){  // 普通文本数据  String fieldName = item.getFieldName(); // 表单元素名称  String content = item.getString(); // 表单元素名称， 对应的数据  //item.getString("UTF-8"); 指定编码  System.*out*.println(fieldName + " " + content);  }  // 上传文件(文件流) ----> 上传到upload目录下  **else** {  // 普通文本数据  String fieldName = item.getFieldName(); // 表单元素名称  String name = item.getName(); // 文件名  String content = item.getString(); // 表单元素名称， 对应的数据  String type = item.getContentType(); // 文件类型  InputStream in = item.getInputStream(); // 上传文件流    /\*  \* 四、文件名重名  \* 对于不同用户readme.txt文件，不希望覆盖！  \* 后台处理： 给用户添加一个唯一标记!  \*/  // a. 随机生成一个唯一标记  String id = UUID.*randomUUID*().toString();  // b. 与文件名拼接  name = id +"#"+ name;    // 获取上传基路径  String path = getServletContext().getRealPath("/upload");  // 创建目标文件  File file = **new** File(path,name);    // 工具类，文件上传  item.write(file);  item.delete(); //删除系统产生的临时文件    System.*out*.println();  }    }    }  **else** {  System.*out*.println("当前表单不是文件上传表单，处理失败！");  }  } **catch** (Exception e) {  e.printStackTrace();  }      } |

### 文件上传与下载，完整案例：

步骤：

1. 文件上传

2. 列表下载

|  |
| --- |
| Index.jsp |
| <body>  <a href=*"*${pageContext.request.contextPath }*/upload.jsp"*>文件上传</a> &nbsp;&nbsp;&nbsp;  <a href=*"*${pageContext.request.contextPath }*/fileServlet?method=downList"*>文件下载</a>    </body> |
| Upload.jsp |
| <body>  <form name=*"frm\_test"* action=*"*${pageContext.request.contextPath }*/fileServlet?method=upload"* method=*"post"* enctype=*"multipart/form-data"*>  <%--<input type="hidden" name="method" value="upload">--%>    用户名：<input type=*"text"* name=*"userName"*> <br/>  文件： <input type=*"file"* name=*"file\_img"*> <br/>    <input type=*"submit"* value=*"提交"*>  </form>  </body> |
| Downlist.jsp |
| <body>  <table border=*"1"* align=*"center"*>  <tr>  <th>序号</th>  <th>文件名</th>  <th>操作</th>  </tr>  <c:forEach var=*"en"* items="${requestScope.fileNames}" varStatus=*"vs"*>  <tr>  <td>${vs.count }</td>  <td>${en.value }</td>  <td>  <%--<a href="${pageContext.request.contextPath }/fileServlet?method=down&..">下载</a>--%>  <!-- 构建一个地址 -->  <c:url var=*"url"* value=*"fileServlet"*>  <c:param name=*"method"* value=*"down"*></c:param>  <c:param name=*"fileName"* value="${en.key}"></c:param>  </c:url>  <!-- 使用上面地址 -->  <a href=*"*${url }*"*>下载</a>  </td>  </tr>  </c:forEach>  </table>  </body> |
| FileServlet.java |
| /\*\*  \* 处理文件上传与下载  \* **@author** Jie.Yuan  \*  \*/  **public** **class** FileServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  // 获取请求参数： 区分不同的操作类型  String method = request.getParameter("method");  **if** ("upload".equals(method)) {  // 上传  upload(request,response);  }    **else** **if** ("downList".equals(method)) {  // 进入下载列表  downList(request,response);  }    **else** **if** ("down".equals(method)) {  // 下载  down(request,response);  }  }      /\*\*  \* 1. 上传  \*/  **private** **void** upload(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    **try** {  // 1. 创建工厂对象  FileItemFactory factory = **new** DiskFileItemFactory();  // 2. 文件上传核心工具类  ServletFileUpload upload = **new** ServletFileUpload(factory);  // 设置大小限制参数  upload.setFileSizeMax(10\*1024\*1024); // 单个文件大小限制  upload.setSizeMax(50\*1024\*1024); // 总文件大小限制  upload.setHeaderEncoding("UTF-8"); // 对中文文件编码处理  // 判断  **if** (upload.*isMultipartContent*(request)) {  // 3. 把请求数据转换为list集合  List<FileItem> list = upload.parseRequest(request);  // 遍历  **for** (FileItem item : list){  // 判断：普通文本数据  **if** (item.isFormField()){  // 获取名称  String name = item.getFieldName();  // 获取值  String value = item.getString();  System.*out*.println(value);  }  // 文件表单项  **else** {  /\*\*\*\*\*\*\*\* 文件上传 \*\*\*\*\*\*\*\*\*\*\*/  // a. 获取文件名称  String name = item.getName();  // ----处理上传文件名重名问题----  // a1. 先得到唯一标记  String id = UUID.*randomUUID*().toString();  // a2. 拼接文件名  name = id + "#" + name;    // b. 得到上传目录  String basePath = getServletContext().getRealPath("/upload");  // c. 创建要上传的文件对象  File file = **new** File(basePath,name);  // d. 上传  item.write(file);  item.delete(); // 删除组件运行时产生的临时文件  }  }  }  } **catch** (Exception e) {  e.printStackTrace();  }      }    /\*\*  \* 2. 进入下载列表  \*/  **private** **void** downList(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    // 实现思路：先获取upload目录下所有文件的文件名，再保存；跳转到down.jsp列表展示    //1. 初始化map集合Map<包含唯一标记的文件名, 简短文件名> ;  Map<String,String> fileNames = **new** HashMap<String,String>();    //2. 获取上传目录，及其下所有的文件的文件名  String bathPath = getServletContext().getRealPath("/upload");  // 目录  File file = **new** File(bathPath);  // 目录下，所有文件名  String list[] = file.list();  // 遍历，封装  **if** (list != **null** && list.length > 0){  **for** (**int** i=0; i<list.length; i++){  // 全名  String fileName = list[i];  // 短名  String shortName = fileName.substring(fileName.lastIndexOf("#")+1);  // 封装  fileNames.put(fileName, shortName);  }  }    // 3. 保存到request域  request.setAttribute("fileNames", fileNames);  // 4. 转发  request.getRequestDispatcher("/downlist.jsp").forward(request, response);  }    /\*\*  \* 3. 处理下载  \*/  **private** **void** down(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    // 获取用户下载的文件名称(url地址后追加数据,get)  String fileName = request.getParameter("fileName");  fileName = **new** String(fileName.getBytes("ISO8859-1"),"UTF-8");    // 先获取上传目录路径  String basePath = getServletContext().getRealPath("/upload");  // 获取一个文件流  InputStream in = **new** FileInputStream(**new** File(basePath,fileName));    // 如果文件名是中文，需要进行url编码  fileName = URLEncoder.*encode*(fileName, "UTF-8");  // 设置下载的响应头  response.setHeader("content-disposition", "attachment;fileName=" + fileName);    // 获取response字节流  OutputStream out = response.getOutputStream();  **byte**[] b = **new** **byte**[1024];  **int** len = -1;  **while** ((len = in.read(b)) != -1){  out.write(b, 0, len);  }  // 关闭  out.close();  in.close();      }    **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  **this**.doGet(request, response);  }  } |

# 2. JavaMail邮件开发

(api + 配置)

开发中，邮件的应用？

-🡪 注册，填写生日； 后期系统会自动发送生日祝贺

-🡪 发货，发货提醒！邮件提醒！

邮件：

1. 发邮件；【程序中如何发邮件！】

2. 收邮件；【很多客户端工具， foxmail 】

## 邮件开发准备:

准备工作, 环境搭建：

1. 本地搭建一个邮件服务器

-🡪 易邮服务器，eyoumailserversetup.exe

2. 新建邮箱账号

张三给李四发邮件。

步骤1：

新建域名： 工具， 服务器设置， 单域名框中输入 itcast.com

步骤2：

新建邮箱账号: [zhangsan@itcast.com](mailto:zhangsan@itcast.com)

[lisi@itcast.com](mailto:lisi@itcast.com)

3. 安装foxmail

配置邮件发送服务器(smtp)： localhost 25

邮件接收服务器(pop3)： localhost 110

再新建账号，就可以接收邮件了！

## 邮件开发之普通邮件:

JavaMail开发，先引入jar文件：

activation.jar 【如果使用jdk1.6或以上版本，可以不用这个jar文件】

mail.jar 【邮件发送核心包】

|  |
| --- |
| /\*\*  \* 1. 发送一封普通邮件  \* **@author** Jie.Yuan  \*  \*/  **public** **class** App\_SendMail {  @Test  **public** **void** testSend() **throws** Exception {    //0. 邮件参数  Properties prop = **new** Properties();  prop.put("mail.transport.protocol", "smtp"); // 指定协议  prop.put("mail.smtp.host", "localhost"); // 主机 stmp.qq.com  prop.put("mail.smtp.port", 25); // 端口  prop.put("mail.smtp.auth", "true"); // 用户密码认证  prop.put("mail.debug", "true"); // 调试模式    //1. 创建一个邮件的会话  Session session = Session.*getDefaultInstance*(prop);  //2. 创建邮件体对象 (整封邮件对象)  MimeMessage message = **new** MimeMessage(session);  //3. 设置邮件体参数:  //3.1 标题  message.setSubject("我的第一封邮件 ");  //3.2 邮件发送时间  message.setSentDate(**new** Date());  //3.3 发件人  message.setSender(**new** InternetAddress("zhangsan@itcast.com"));  //3.4 接收人  message.setRecipient(RecipientType.*TO*, **new** InternetAddress("lisi@itcast.com"));  //3.5内容  message.setText("你好，已经发送成功！ 正文...."); // 简单纯文本邮件  message.saveChanges(); // 保存邮件(可选)    //4. 发送  Transport trans = session.getTransport();  trans.connect("zhangsan", "888");  // 发送邮件  trans.sendMessage(message, message.getAllRecipients());  trans.close();  }  } |