# 上传文件

## 上传页面不刷新

### 前端

原理方式：注意：ajax是不支持文件上传的

1. 使用iframe标签关联form表单：iframe标签用name属性值和form标签的target属性值一一对应
2. 设置iframe的框为不显示：style="display: none;"
3. Form进行上传文件该有的还要设置 如：enctype="multipart/form-data"和 <input type="file">

这3步就实现了无刷新页面效果

<body>

<iframe style="display: none;" name="abc"></iframe> 页面刷新其实就是这个页面框，设置了隐藏所以看不到刷新效果

<form target="abc" method="post" action="xxx" enctype="multipart/form-data">

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

<input type="submit" name="upload" />

</form>

</body>

使用jquery插件：OCUpload进行操作：它就是跟原理方式效果实现一样的

jquery.ocupload-1.1.2.js就这一个js文件

使用步骤

第一步 HTML只要定义一个：<input id="eee" type="button" value="上传 " />即可，没有特殊的要求

第二步方法：$("input[name=ddd]")..upload({

name: 'file',//上传组件的name属性，即<input type='file' name='file'/>

action: '',//向服务器请求的路径

enctype: 'multipart/form-data',//mime类型，默认即可

params: {},//请求时额外传递的参数，默认为空

autoSubmit: true,//是否自动提交，即当选择了文件，自动关闭了选择窗口后，是否自动提交请求。

onSubmit: function() {},//提交表单之前触发事件

onComplete: function() {},//提交表单完成后触发的事件

onSelect: function() {}//当用户选择了一个文件后触发事件

});

EasyUI方法

$('#ff').form('submit')

# 下载

Web技术19天销售绑定下载

public class SaleListServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

//1.调用Service中查询销售榜单的方法

OrderService service = BasicFactory.getFactory().getService(OrderService.class);

List<SaleInfo> list = service.saleList();

//2.将销售榜单信息组织称csv格式的数据

StringBuffer buffer = new StringBuffer();

buffer.append("商品编号,商品名称,销售数量\r\n");

for(SaleInfo si : list){

buffer.append(si.getProd\_id()+","+si.getProd\_name()+","+si.getSale\_num()+"\r\n");

}

String data = buffer.toString();

//3.提供下载

String filename = "Estore销售榜单\_"+new Date().toLocaleString()+".csv"; //csv格式文件和excel差不多(可以普通文本打开，**逗号分隔值**)

response.setHeader("Content-Disposition", "attachment;filename="+URLEncoder.encode(filename,"utf-8"));//设置下载头

response.setContentType(this.getServletContext().getMimeType(filename));//配合用，设置下载的默认文件名

response.getWriter().write(data);//

}

# 权限管理

## 原始方法

简单介绍 (web技术视频第19天) 注意：虽然low但是给你一个配置权限方向感

1使用filter(进行权限管理)

2使用txt文本每一行记录uri最后的地址：如 localhost:8080/wwl/user.jsp,这个user.jsp就是要记录的内容

3使用字符缓冲流一行一行读取文本txt内容封装到list中

使用步骤

第一步 创建filter过滤器

public class PrivilegeFilter implements Filter {

//封装从txt文件中读取的内容：每一个list代表不同权限文件中的内容

private List<String> admin\_list = new ArrayList<String>();//admin权限

private List<String> user\_list = new ArrayList<String>();//普通用户权限能访问的页面

public void destroy() {}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

HttpServletRequest req = (HttpServletRequest) request;

HttpServletResponse resp = (HttpServletResponse) response;

String uri = req.getRequestURI();//获取用户访问页面的路径uri

if(admin\_list.contains(uri) || user\_list.contains(uri)){

//说明当前资源需要权限：查看用户有无登录

if(req.getSession(false)==null || req.getSession().getAttribute("user")=="null"){

response.getWriter().write("当前资源需要权限,请先登录!");

return;

}

User user = (User) req.getSession().getAttribute("user");

//判断用户请求的页面是否有权限：contains方法是判断是否有此内容

if(admin\_list.contains(uri) && "admin".equals(user.getRole())){

chain.doFilter(request, response);//放行

}else if(user\_list.contains(uri) && "user".equals(user.getRole())){

chain.doFilter(request, response);

}else{

throw new RuntimeException("您不具有对应的权限!!!");

}

}else{

//不需要权限：一般就是登录页面和注册页面

chain.doFilter(request, response);

}

}

//过滤器在启动web容器的时候就进行加载权限配置文件：放入list中(内存中)

public void init(FilterConfig filterConfig) throws ServletException {

ServletContext context = filterConfig.getServletContext();

try {

//加载admin.txt

BufferedReader adminReader = new BufferedReader(new FileReader(context.getRealPath("WEB-INF/admin.txt")));

String line = null;

while((line=adminReader.readLine())!=null){

admin\_list.add(line);

}

//加载user.txt

BufferedReader userReader = new BufferedReader(new FileReader(context.getRealPath("WEB-INF/user.txt")));

line = null;

while((line=userReader.readLine())!=null){

user\_list.add(line);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

第二步 配置文件内容 其实老师建议放入数据库中

Admin.txt下的内容：以下内容只有admin管理员可以访问

/addProd.jsp

/AddprodServlet

/SaleListServlet

user.txt内容 以下页面普通用户可以访问

/addOrder.jsp

/cart.jsp

/confirm.jsp

/orderList.jsp

/pay.jsp

/AddCartServlet

/ChangeCartServlet

/ClearCartServlet

/DelCartServlet

/DelOrderServlet

/OrderListServlet

/PayServlet

## Shiro

https://blog.csdn.net/jinnianshilongnian/article/details/84528037 详细的shiro使用

### 使用步骤

第一步 导包

除了基本的以外，我们还需要shiro-web和shiro-spring的的架包，下面是所需要的所有shiro架包，至于其他的架包，像缓存的架包，Spring和SpringMVC的架包还是和我们以前使用的架包一样的。

核心包：shiro-core-1.2.2.jar、commons-beanutils-1.8.3.jar

LOG包： slf4j-api-1.6.4.jar、commons-logging-1.1.3.jar

web工程包: shiro-web

spring整合包: shiro-spring

Maven

<!-- 日志包：commons-logging -->

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.3</version>

</dependency>

<!-- 核心包：有三个文件： shiro-core、commons-beanutils、slf4j-api、 -->

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-core</artifactId>

<version>1.3.2</version>

</dependency>

<!-- web工程包 ：shiro-web-->

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-web</artifactId>

<version>1.2.3</version>

</dependency>

<!-- shiro和spring整合包 -->

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-spring</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

第二步 web.xml配置

<