

# 1.我们每次新建 servlet的时候千万不要直接复制一个，我们最好是新建一个，因为如果是复制一个的话，不会web.xml中 产生路径的

# 2.我想的到主机名,消息头

2.我想的到主机名ip req.getRemoveteAddr();

机器名 req.getRemoveteHost();

<%// out.println(request.getRemoteHost());

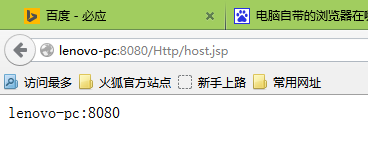
out.println(request.getHeader("host"));

%>

<br/>

<% //out.println(request.getRemoteAddr());

%>





# 3.有的时候我们不希望用户直接跳到我们中间的界面，使用浏览器获取referer

## 1.如下，中间的界面内容如下

<body>

<%//获取浏览器的referer

String referer = request.getHeader("referer");

out.println(referer);

**if**(referer==**null**){

response.sendRedirect("error");

}

out.println("这里面的东西你看不到");

%>

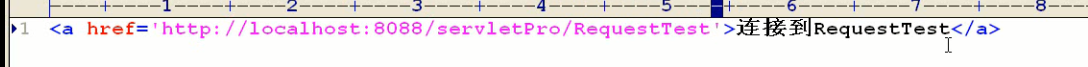
## 2.如果用户想要直接到了这个位置，那么我们就将他跳转到了error界面

<body>

你是非法盗链者

</body>

## 3.但是我们还有可能是通过超链接过来，我们上面的是直接输入的网址，所以使用referer给阻拦下了，但是如果是使用超链接的话话，也是可以的阻拦的



## 4. referer上添加手脚我们如果想让用户必须是通过我们的自己的网站过来的，可以再，这样我们的referer是真的有了值了 ，这样就是保证我们从我们自己的网站过来（是不是和我们的session制作用户的登录有点像）只要是从自己的网站过来就会给referer赋值了

<%//获取浏览器的referer

String referer = request.getHeader("Referer");

out.println(referer);

**if**(referer==**null**||

!referer.startsWith("http://lenovo-pc:8080/Http")){

response.sendRedirect("/Http/error.jsp");

}

out.println("这里面的东西你看不到");

%>

## 4.2对上面的东西进行下测试，新建一个jsp文件，tiao.jsp

<body>

<a href=*"http://lenovo-pc:8080/Http/referer.jsp"*>跳转到referer界面</a>

</body>

## 4.3这样我们的界面是可以跳转到http上面的

## 4.4这样我们的防盗链就制作完成了

# 4正确使用sendRedirect



## 4.1上面的代码等价于



# 5.http响应消息头详解



## 5. refresh15秒钟后跳转到另一个界面，现在我们要制作衣下里面的饿，

<%

response.setHeader("Refresh", "5;url=/Http/tiao.jsp");

%>

<%

out.println("看看5秒后发生了什么");

%>

## 5.2告诉浏览器有文件下载



### 1. Content-Disposition现将一个图片复制到我们的images文件夹下面，注意下面的图片的地址，一定要是相对的路径

<%

//测试Content-Disposition

//演示下载文件

response.setHeader("Content-Disposition", "attachment;filename=1.png");

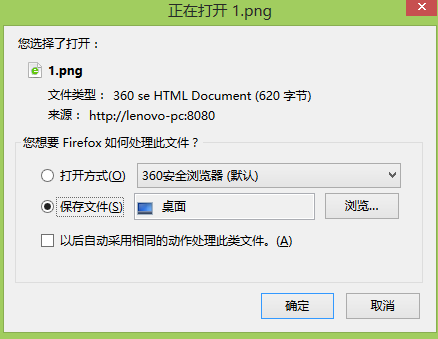
//打开文件，说明下web下载文件的原理

//1.获取要下载文件的全路径

String path1 = **this**.getServletContext().getRealPath("/images/1.png");

%>

### 2.然后就会出现下面的东西



<%

//测试Content-Disposition

//演示下载文件

response.setContentType("text/html;charset=utf-8");

response.setHeader("Content-Disposition", "attachment;filename=1.png");

//打开文件，说明下web下载文件的原理

//1.获取要下载文件的全路径

String path1 = **this**.getServletContext().getRealPath("/images/1.png");

System.out.println(path1);

//下面就是打开这个文件(首先要读出了)

FileInputStream fis = **new** FileInputStream(path1);

// 通过文件输入流写入文件

OutputStream oStream = response.getOutputStream();

**byte** buffer[] = **new** **byte**[255555];

**int** len =0; //表示实际每次读取了多少个字节

**if**((len=fis.read(buffer))>0){

oStream.write(buffer,0,len);

}

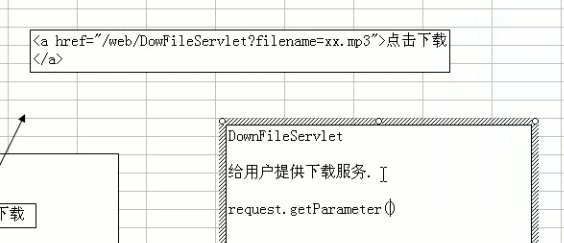
oStream.close();

fis.close();

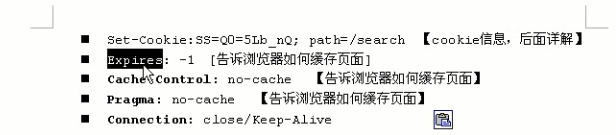
%>

### 3.这样就完成了，我们可以设置下载各种的东西

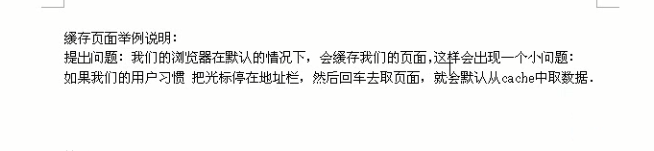
只要将上面的 图片名字改变就可以了



# 6.缓存



## 1.缓存界面举例说明，浏览器默认情况下，会默认的缓存我们的界面

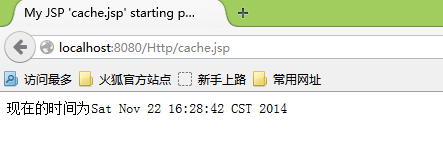


## 2. Cache-control有的时候我们必须保证不能缓存，比如基金网站import=*"java.util.\*"*，

<%

out.println("现在的时间为"+**new** Date());

%>



## 3.现在就是开始设置不能够缓存了,将来我们做一些大的网站，一定要注意缓存的设置

<% //这个是ie浏览器,指定下面的页面不缓存，下面的是 -1的时候就是设置了不缓存

response.setDateHeader("Expeires", -1);

//为了保证兼容性，上面的那个是时间上的设置缓存，下面的这个就不是完全不缓存了

response.setHeader("Cache-control", "no-cache");

response.setHeader("Pragma", "no-cache");

out.println("现在的时间为"+**new** Date());

%>

## 4.有的时候我们希望上面的就是缓存一个一定的时间

<% //这个是ie浏览器,指定下面的页面不缓存，下面的是 -1的时候就是设置了不缓存

//获取一个当前的一个小时的毫秒

response.setDateHeader("Expeires", System.currentTimeMillis()+3600\*1000\*24);  **缓存一天**

//为了保证兼容性，上面的那个是时间，下面的这个就不是时间了

// response.setHeader("Cache-control", "no-cache");

//response.setHeader("Pragma", "no-cache");

out.println("现在的时间为"+**new** Date());

%>

