* Servlet的细节问题

由于客户端是通过url地址访问web服务器中的资源的，所以servlet程序若想被外界访问，必须把servlet程序映射到一个URl地址上，这个工作在web.xml文件中使用<servlet>元素和<servlet-mapping>元素完成。

<servlet>元素主要用来设置servlet注册的名称<servlet-name>和servlet完整的类路径<servlet-class>，一个注册好的servlet可以被多次映射。

<servlet-mapping>元素主要用来制定servlet的注册名称<servlet-name>和servlet的对外访问路径<url-pattern>，它是对一个已经注册的servlet映射，他们不是一一对应的关系。

①：一个已经被注册的servlet可以被多次映射

②：当映射一个servlet的时候，可以多层映射 比如

<servlet>

<description>This is the description of my J2EE component</description>

<display-name>This is the display name of my J2EE component</display-name>

<servlet-name>Myservlet1</servlet-name>

<servlet-class>com.wsx.servlet.Myservlet1</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Myservlet1</servlet-name>

<url-pattern>/Myservlet1</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Myservlet1</servlet-name>

<url-pattern>/da.html</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Myservlet1</servlet-name>

<url-pattern>/servlet/12.servlet</url-pattern>

</servlet-mapping>

从这里还可以看出，后缀名是.html不一定就是html文件

③：在对servlet进行映射的时候，可以使用通配符号

有两种格式可以使用：

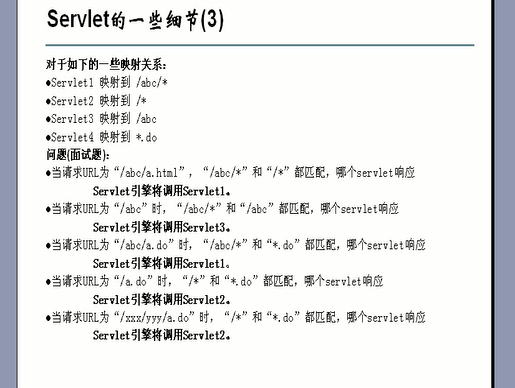
第一种，\*.扩展名 比如 \*.do，\*.jss

第二种，以“/”开头,同时以“/\*”结尾 比如/\* /news/\*

在匹配的时候，要参考两个标准:

第一步：谁的匹配度高，谁就被选中

第二步：\*.do的优先级别最低



如何使网站被关闭，如何配置servlet-mapping?

将url-pattern中写入”/\*”

④：servlet的单例问题：

当servlet被访问后，就被加载到内存，以后该实例就对各个请求服务，即在使用中是单例的

因为servlet是单例的，会出现一个线程安全问题：比如

售票系统 如果不加同步机制，则可能会出现问题

给大家一个原则：

1. 如果一个变量需要多个用户共享，则应当在访问该变量的时候加同步机制

Synchronized (对象){

//同步代码

}

1. 如果一个变量不需要共享，则直接在doGet()或者doPost()中定义，这样也不会存在线程安全问题。

⑤：servlet 中的<load-on-startup>配置

需求：当我们的网站启动的时候，可能会要求一些初始化的数据，(比如创建临时表)，再比如，我们的网站有一些要求定时完成的任务【定时备份数据库，定时发送邮件】

解决方法：可以通过<load-on-startup>配合线程知识搞定

先说明<load-on-startup>：通过配置<load-on-startup> 我们可以指定某个Servlet自动创建。

我们可以来模拟一个定时发送电子邮件的功能

实现思路

SendEmailTable

ID content sendtime

1 “hello” 2011-11-13 20:00

2 “hello2” 2012-11-11 08:00

看线程如何去完成任务

这里的代码请参考项目servletPro

SendMail.java

**package** com.wsx.model;

**public** **class** SendMail **extends** Thread{

@Override

**public** **void** run() {

**int** i =0;

**try** {

**while**(**true**){

//休眠10秒

Thread.*sleep*(10\*1000);

//发出邮件

System.*out*.println("发出第"+(++i)+"封邮件");//javamail

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

}

}

MyinitServlet1.java

的init函数

**public** **void** init() **throws** ServletException {

// Put your code here

System.*out*.println("MyinitServlet1 被调用...");

//加载到网页中去

SendMail sendmail =**new** SendMail();

sendmail.start();

}

说明：

<!-- 1 表示servlet被init的顺序-->

<load-on-startup>1</load-on-startup>

代表启动的顺序

* ServletConfig对象

该对象主要用于读取servlet的配置信息。

<servlet>

<description>This is the description of my J2EE component</description>

<display-name>This is the display name of my J2EE component</display-name>

<servlet-name>MyServletConfigTest</servlet-name>

<servlet-class>com.wsx.servlet.MyServletConfigTest</servlet-class>

<!-- 这里配置的servlet信息只能被该servlet读取 -->

<init-param>

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

<param-value>UTF-8</param-value>

</init-param>

</servlet>

如何使用：

response.setContentType("text/html");

String encoding =**this**.getServletConfig().getInitParameter("encoding");

response.setCharacterEncoding(encoding);

PrintWriter out = response.getWriter();

out.print("编码是："+encoding);

补充说明一下：这种配置参数的方式，只能被某个servlet独立使用

如果希望让所有的servlet都能读取某个参数，这样配置：

<!-- 这个是类似于全局变量的 参数，可以被所有servlet读取 -->

<context-param>

<param-name></param-name>

<param-value></param-value>

</context-param>

* 如果想把所有的参数都读取，则使用如下方法:

Enumeration<String>names=**this**.getServletConfig().getInitParameterNames();

**while**(names.hasMoreElements()){

String name=names.nextElement();

System.*out*.println(name);

System.*out*.println(**this**.getServletConfig().getInitParameter(name));

}

* 接下来开始开发

用户登录界面：

1. 先完成用户登陆
2. 在主界面添加一个超链接，可以返回登录界面重新登陆

* http协议的再介绍

1.http协议是建立在tcp/ip协议基础之上的

2.http协议全称是 超文本传输协议

3.http协议 1.0版本 1.1版本，目前通用的是1.1版本

http1.0 短连接 结束后直接断开连接

http1.1长连接 会保持一段时间

长短说的是连接持续的时间 1.1 30s左右

* http的请求部分：

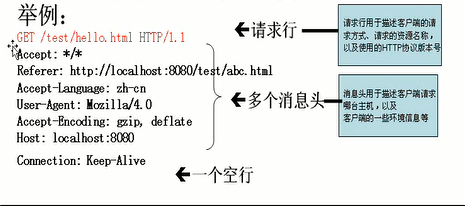
基本结构：

请求行

消息头 （每次请求不同，内容也不同）

（空行）

发送的内容【格式：内容名字=内容体】



Get和post的区别：参考之前笔记

* 详解http请求消息头

1. Accept：text/html,image/\* [告诉浏览器，我可以接收文本网页图片]
2. Accept-Charset :ISO-8859-1 [接受字符编码：ISO-8859-1]
3. Accept-Language:en-us,zh-cn[浏览器支持的语言]
4. Accept-Encoding:gzip,Compress[可以接收gzip，compress压缩后的数据]
5. Host:www.sohu.com [要找的主机是谁]
6. If-Modified-Since:Tue,11 Jul 2000 18:23:51 GMT

[告诉服务器我的缓存中有这个资源文件，该文件的时间是 如上 。。。]

1. Referer:http://www.sohu.com/index.com[告诉服务器我来自哪里，该消息头常用于防止盗链]
2. User-Agent:Mozilla/4.0 [这个浏览器的内核是]
3. Cookie []
4. Connection:close/Keep-Alive [发完数据后不断开链接]
5. Date:Tue,11 Jul 2000 18:23:51 GMT[浏览器发送这个http请求时的时间]

Referer防盗链实际案例:

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

out.print("Referer="+referer);

**if**(referer==**null**||!referer.startsWith("http://lenovo-pc:8080/servletPro")){

response.sendRedirect("/servletPro/Error.html");

}

String host=request.getHeader("Host");

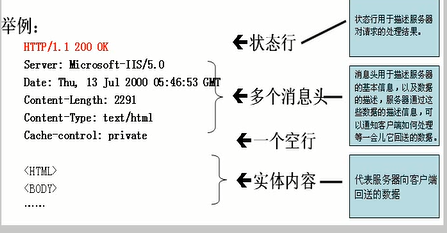
out.print("<br>");

out.print("host="+host);

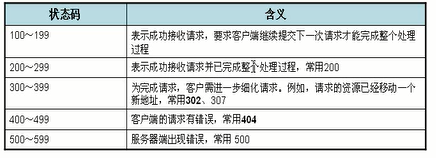
out.print("账户密码:12345679");

红色部分可以根据实际情况来修改

* 详解http响应



* http 响应的状态行的举例说明:



200 就是整个请求和响应过程没有发生生错误,这个最常见

302 当你请求一个资源的时候,服务器返回302表示,让浏览器转向另外一个资源,

比如:Response.sendRedirect(“/web应用名/资源名”)

例子:

response.setStatus(302);

response.setHeader("Location", "http://www.baidu.com");

//上面两句话等价于

response.sendRedirect("/servletPro/Servlet2");

404 代表找不到资源的错误

500 服务器端出现错误 internal error

* http响应消息头的详解:

Location:http://www.baidu.com [让浏览器重新定位url]

Server:Apache tomcat [告诉浏览器我是tomcat]

Content-Encoding:gzip [告诉浏览器我用了gzip]

Content-Length:80[告诉浏览器回送的数据大小80字节]

Content-Language:zh-cn[支持中文]

Content-Type:text/html;charset=gb2312[内容格式text/html;编码是gb2312]

Last-Modified:Sat, 04 Feb 2017 04:18:02 GMT[告诉浏览器这个资源上一次修改时间是什么时候]

Refresh:1;url=http://www.baidu.com [过多久,刷新到http://www.baidu.com]

Content-Disposition:attachment;filename=aaa.zip[告诉浏览器,有文件下载]

Transfer-Encoding:chunked[传输的编码]

Set-Cookie:SS=Q0=5Lb\_nQ;path=/search [cookie信息,后面详解]

Expires:-1[告诉浏览器如何缓存页面][ie识别的]

Cache-Control:no-cache[告诉浏览器如何缓存页面][Firefox识别的]

Pragma:no-Cache[告诉浏览器如何缓存页面]

Connection:close/Keep-Alive

Date:Sat, 04 Feb 2017 04:18:02 GMT

Content-Disposition例子:

//演示下载文件

response.setHeader("Content-Disposition", "attachment;filename=123.jpg");

//打开文件说明一下web站点下载文件的原理

//获取到要下载文件的全路径

String path=**this**.getServletContext().getRealPath("/images/123.jpg");

//System.out.println("path="+path);

//创建文件输入流

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

//做一个缓存字节数组

**byte** buff[]=**new** **byte**[1024];

**int** len=0;//表示实际每次读取多少字节

OutputStream os=response.getOutputStream();

**while** ((len=fis.read(buff))>0){

os.write(buff, 0, len);

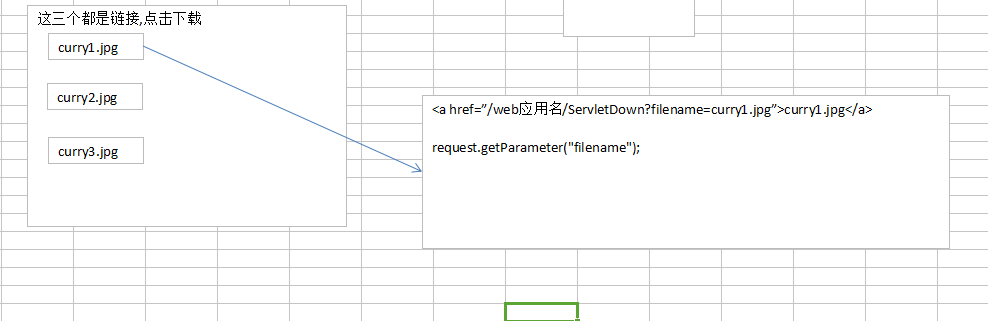
}

//关闭流

os.close();

fis.close();

文件下载练习:



response.setContentType("text/html");

//PrintWriter out = response.getWriter();

String filename=request.getParameter("filename");

//System.out.println(filename);

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

//System.out.println(response.getHeader("Content-Disposition"));

String path=**this**.getServletContext().getRealPath("/images/"+filename);

//System.out.println(path);

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

**int** len=0;

**byte** buff[]=**new** **byte**[1024];

OutputStream os=response.getOutputStream();

**while**((len=fis.read(buff))>0){

os.write(buff, 0, len);

}

os.close();

fis.close();

缓存页面举例说明:

提出问题:我们的浏览器在默认情况下,会缓存我们的页面.这样就会产生一个小问题,如果我们的用户习惯通过把光标停在地址栏回车去去页面,就会默认从Cache中去取数据

1. 有些网站对即时性要求很高,因此要求我们不缓存页面

代码:

out.print(**new** java.util.Date());

//供IE识别

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

//为了保证兼容性

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

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

1. 有些网站要求网页缓存一定时间即可,比如缓存一个小时

response.setDateHeader("Expires", System.*currentTimeMillis*()+3600\*1000);