# 缺省的servlet（了解）

## 自己编写的servlet，不要配置/（斜杠）

## 在tomcat/conf/web.xml中的配置。

<servlet>

<servlet-name>default</servlet-name>

<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>0</param-value>

</init-param>

<init-param>

<param-name>listings</param-name>

<param-value>true</param-value>

</init-param>

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

</servlet>

# response对象（响应对象）

## 想要获取客户端的内容，使用request对象。对客户端做出响应使用response对象。

## 响应：

### 响应行

#### 状态码：

##### public void setStatus(int sc); 设置状态码

### 响应头[key:value的形式，一个key对应一个value，一个key对应多个value]

##### 设置头信息

**public** **void** setHeader(String name, String value); [一个key对应一个value] 经常使用的

response.setHeader("name", "value");

response.setHeader("name", "value1");

结果：name: "value1"

**public** **void** setIntHeader(String name, **int** value);

**public** **void** setDateHeader(String name, **long** date); 值是毫秒值[int 秒 long 毫秒]

**public** **void** addHeader(String name, String value); [一个key对应多个value]

response.addHeader("name", "value");

response.addHeader("name", "value1");

结果：name: "value", "value1"

**public** **void** addIntHeader(String name, **int** value);

**public** **void** addDateHeader(String name, **long** date);

### 响应体

ServletOutputStream getOutputStream() 字节输出流

PrintWriter getWriter() 字符输出流

## 案例

### 重定向（登陆页面）

#### 登陆页面重定向

#### 状态码302 ,应头location

#### 需求：登陆页面，用户名和密码的登陆页面，用户名和密码都是admin,如果有一个不是，重定向到登陆页面，重新登陆。

// 设置302的状态码

response.setStatus(302);

// 设置地址

response.setHeader("location", "/day10/response/login.html");

// response对象提供了一个方法，可以完成重定向。

response.sendRedirect("/day10/response/login.html");

### 页面定时刷新（页面读秒操作）

#### 响应的头 refresh

<meta http-equiv="refresh" content="5;url=/day10/response/login.html">

### 禁用浏览器缓存（三个头信息）

#### 应用：网银。页面的数据是发送变化的

Cache-Control : no-cache

Expires: -1 值是日期类型（setDateHeader()）

Pragma : no-cache

### 向页面输出中文（乱码问题）

#### 字节：ServletOutputStream getOutputStream() , 字节输出流

##### 字节的输出中文的乱码

###### 输出“哈罗我的”是否乱码呢？

不一定乱码。

解决办法

设置浏览器打开文件时所采用的编码

esponse.setHeader("Content-Type", "text/html;charset=UTF-8");

获取字符串byte数组时编码和打开文件时编码一致。

"哈罗我的".getBytes("UTF-8")

#### PrintWriter getWriter() ，字符输出流

##### 字符输出中文是否乱码呢？

###### 肯定乱码，原因如下

response缓冲区的编码，默认值ISO-8859-1

###### 设置response缓冲编码

response.setCharacterEncoding("UTF-8");

##### 设置浏览器打开文件所采用的编码

response.setHeader("Content-Type", "text/html;charset=UTF-8");

##### 简写方式

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

#### 总结：response对象输出中文，产生乱码。

##### 字节

###### 解决方案

设置浏览器打开文件时采用的编码

response.setHeader("Content-Type", "text/html;charset=UTF-8");

获取字符串的byte数组采用的编码

"哈罗我的".getBytes("UTF-8")

##### 字符

###### 解决方法

设置浏览器打开文件时采用的编码

response.setHeader("Content-Type", "text/html;charset=UTF-8");

设置response缓冲区的编码

response.setCharacterEncoding("UTF-8");

###### 简写的方式（等于上面的两句）

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

### 实现文件下载（初级的下载）

#### 超链接，浏览器可以解析直接打开

##### 弹出下载窗口

###### 前提条件：先准备一个文件

###### 读取文件，读入到输入流中

###### 通过response读出到浏览器端

###### 设置头信息

Content-Disposition attachment;filename=文件名称

以附件的形式打开。

### 实现验证码（\*\*\*\*\*）

# request对象（请求对象）

* ServletRequest
  + HttpServletRequest

## ·获取客户机信息

getRemoteAddr（\*\*\*\*\*） 获取IP地址

getMethod() （\*\*\*\*\*） 获取请求方式

getContextPath()（\*\*\*\*\*） 获取虚拟路径

## 获取请求头信息

String getHeader(String name)

long getDateHeader(String name)

int getIntHeader(String name)

### 请求头

referer 记住当前网页的来源

User-Agent 判断浏览器

if-modified-since 控制缓存

## 获取请求参数（\*\*\*\*\*）

String getParameter(String name) （\*\*\*\*\*）

String[] getParameterValues(String name)（\*\*\*\*\*）

Map getParameterMap()（\*\*\*\*\*）

Enumeration getParameterNames()（用的比较少）

### 乱码问题解决：

#### POST请求乱码 ：

request.setCharacterEncoding("utf-8");

#### GET请求乱码

##### 解决方案一：修改tomcat/conf/server.xml

<Connector port="80" protocol="HTTP/1.1"

connectionTimeout="20000"

edirectPort="8443" URIEncoding="utf-8"/>

必须有修改tomcat服务器配置文件权限

##### 解决方案二：逆向编解码

username = URLEncoder.encode(username, "ISO8859-1");

username = URLDecoder.decode(username, "utf-8");

##### 解决方案三：简写的方式（推荐使用）

username = new String(username.getBytes("ISO-8859-1"),"utf-8");

##### request获取中文数据乱码（总结：）

###### post提交

设置request缓冲区的编码

request.setCharacterEncoding("utf-8");

###### get提交

String构造方法

username = new String(username.getBytes("ISO-8859-1"),"utf-8");

## 利用请求域传递对象（request域对象）

### 重定向和转发的区别（转发）（\*\*\*\*\*）

#### 域对象

##### ServletContext：服务器一启动，为每个web应用创建一个ServletContext对象，所有servlet实例共享对象。

##### request：一次请求的范围

getAttribute("","");

getAttribute("");

removeAttribute("");

getRequestDispatcher(String path) ，返回是RequestDispatcher：对象

##### RequestDispatcher：

forward(ServletRequest request, ServletResponse response)（经常使用） 转发的方法

include(ServletRequest request, ServletResponse response)（了解） 包含