Response

2018年10月15日 9:25

1. 代表http响应的对象

ServletResponse -- HttpServletResponse

- 继承结构 (!掌握)

  ServletResponse -- 通用的响应接口, 定义了响应对象应该具有的功能

  |

  |--HttpServletResponse 在ServletResponse的基础上, 添加了很多和Http协议相关的方法

- Response上的重要方法

  一个状态行: HTTP/1.1 200 OK

  若干响应头:

  xxx

  ...

  xxx

  (一个空行)

  实体内容: xxxxx

  - *设置状态码的方法*

    void setStatus(int sc)

    void setStatus(int sc, String sm)

  - *设置响应头的方法*

    void setHeader(String name, String value)

    void setDateHeader(String name, long date)

    void setIntHeader(String name, int value)

    void addHeader(String name, String value)

    void addDateHeader(String name, long date)

    void addIntHeader(String name, int value)

  - *设置响应内容的方法*

    ServletOutputStream getOutputStream()

    PrintWriter getWriter()

Response的功能

2018年10月15日 11:22

1. Response的功能 (!掌握)
   - 向客户端发送数据 (!!!重点, 特别是乱码问题)

     getOutputStream()

     getWriter()

     - **字节流发送数据的中文乱码问题

       服务器端指定了用utf-8来发送数据, 浏览器在接受数据时, 如果不指定将使用默认的平台码GBK, 编解码不一致导致乱码.

       - 解决方案:

         response.setHeader("Content-Type", "text/html;charset=utf-8");// 通知浏览器使用utf-8打开服务器发送过去的数据

         response.getOutputStream().write("中国".getBytes("utf-8"));// 指定编码为utf-8
     - 字符流发送数据的中文乱码问题

       利用字符流发送数据, 底层还是要转成字节. 服务器端如果不手动指定, 服务器默认会使用iso8859-1码表, 由于里面没有中文汉字, 所以服务器端发送给客户端就是一堆乱码, 客户端不管使用什么码表都无法转成正常的字符。

       服务器会根据getCharacterEncoding()方法返回的编码来发送数据, 如果没有指定, 该方法默认返回iso8859-1。

       - 解决方案:

         第一步, 需要指定服务器发送数据使用utf-8

         response.setCharacterEncoding("utf-8"); //通知服务器使用utf-8来发送响应实体中数据

         第二步: 需要指定浏览器在接收数据时也使用同一个编码来打开数据

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

         等价于<==>

         response.setContentType("text/html;charset=utf-8");
     - 在通知浏览器使用什么编码接受服务器发送的数据时, 服务器很智能, 会使用相同的编码来发送数据, 所以指定服务器以什么编码发送数据的代码可以省略不写
     - *总结:*

       - 不管是字符流还是字节流, 解决乱码问题, 可以用一行代码搞定:

         response.setContentType("text/html;charset=xxx");
     - getOutputStream()和getWriter() 这两个方法是互斥的, 在一次请求当中调用了其中的一个, 就不能再调用另一个!!!
     - 在调用完getOutputStream()或getWriter()方法之后, 不需要手动去关闭流, 服务器会自动帮我们去关闭!!!
   - 这个两个方法获取到的流并不是指向客户端的流, 而是指向response缓冲区的流, 通过流将数据写入response缓冲区, service方法执行结束, 请求回到服务器, 由服务器将数据组织成响应消息打给浏览器!!
   - 实现重定向 (!!重点)

     重定向的原理就是302+location, 通过设置状态码302和location响应头就可以实现重定向的效果

     response.setStatus(302);

     response.setHeader("Location", "/Day09/index.jsp");

     这两行代码等价于

     response.sendRedirect("/Day09/index.jsp");

两次请求 两次响应，地址会发生变化，重定向不仅可以跳转到当前web应用或者虚拟主机，也可以跳转到其他的web应用或虚拟主机。

- 实现定时刷新 (!掌握)

  定时刷新是通过Refresh响应头, 可以实现在多少秒之后跳转到另外一个资源

  response.setHeader("Refresh", "3;url=/Day09/index.jsp");

三种跳转方式总结

2018年10月15日 11:22

1. 总结: (!掌握)
   - 请求转发/请求重定向/定时刷新都可以实现资源的跳转, 区别是什么呢?
     - 请求转发:

       一次请求,一次响应 request对象是同一个

       地址栏不会发生变化

       只能用于服务器内部的资源跳转, 并且只能是同一应用中的不同资源上进行跳转, 不可用在不同应用和不同服务器中的资源跳转
     - 请求重定向:

       两次请求,两次响应 request对象不是同一个

       地址栏会发生变化

       可以用于服务器内部的资源跳转, 也可以用于不同应用和不同服务器之间的资源跳转
     - 定时刷新:

       两次请求,两次响应 request对象不是同一个

       地址栏会发生变化

       可以用于服务器内部的资源跳转, 也可以用于不同应用和不同服务器之间的资源跳转

       和重定向不同的是, 定时刷新可以在刷新到新的地址之间设置一个时间, 在间隔的这段时间内可以输出文本到浏览器并维系一段时间。
   - 那什么时候用哪种方式进行资源的跳转呢?

     如果是同一服务器中的同一应用内部的资源跳转:

     ~如果需要利用request域在跳转的资源之间传输数据, 只能用请求转发

     ~如果不想让地址栏发生变化, 只能用请求转发

     ~如果需要地址栏发生变化, 只能用重定向或定时刷新

     ~如果没有什么特殊需求, 三种方式都可以, 但是推荐使用转发, 可以减少请求次数降低服务器的压力.

     ~如果只是想更新刷新操作, 最好使用重定向或定时刷新, 使用请求转发, 在刷新时会把刚才的操作再做一遍, 可能会导致一些问题, 比如表单重复提交或重复支付订单等

     如果是不同服务器或不同应用内部的资源跳转, 只能用重定向或这定时刷新:

     重定向和定时刷新的主要区别在于: 重定向会立即跳转, 而定时刷新可以设置一个时间间隔, 在指定时间后再进行跳转.

     如果在跳转之前需要输出提示信息(如: 注册成功, xx秒后跳转到xxx)只能用定时刷新, 否则两种方式都可以.

利用response控制浏览器缓存

2018年10月15日 11:24

1. 控制浏览器的缓存行为 (!掌握)

由于不同的浏览器的缓存行为可能是不同的, 我们可以在服务器中通过设置响应头来控制浏览器的缓存行为!!

控制浏览器不要缓存:

```
response.setDateHeader("Expires", -1);
response.setHeader("Cache-control", "no-cache");
response.setHeader("Pragma", "no-cache");
```

控制浏览器缓存:

```
response.setDateHeader("Expires", System.currentTimeMillis()+1000*60*60*24);    //24小时
response.setHeader("Cache-control", "max-age=60");   //60秒
```