**Experiment3**

1. 编写JSP程序，实现图4-24所示的简易计算器。要求：输入“第一个参数”，选择运算类型（+，-，\*，/），输入 “第二个参数”后，按“计算”按钮，结果将显示在“结果”文本框中。

注意：程序需要对输入参数是否合法进行判断，例如参数是否为数字，除法时，除数不为0的判断。

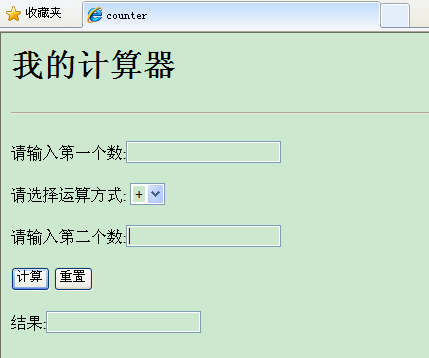
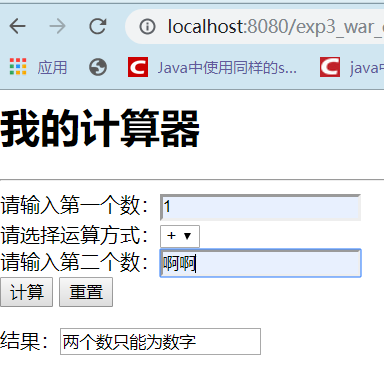
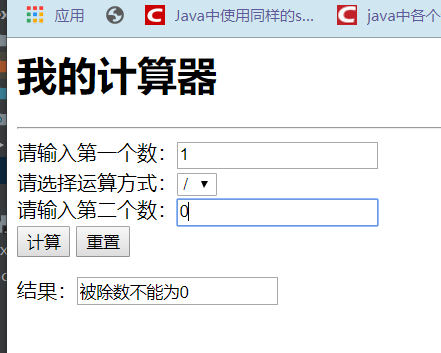
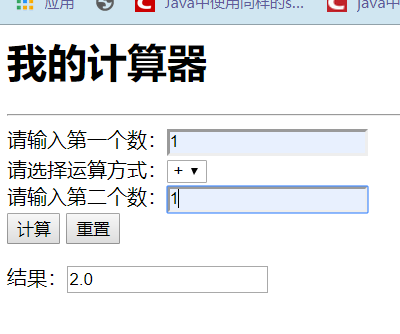


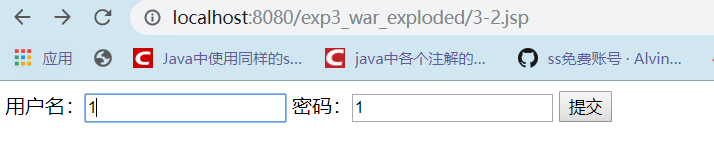
图4-24 简易计算器





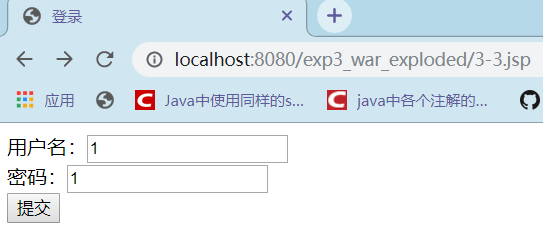
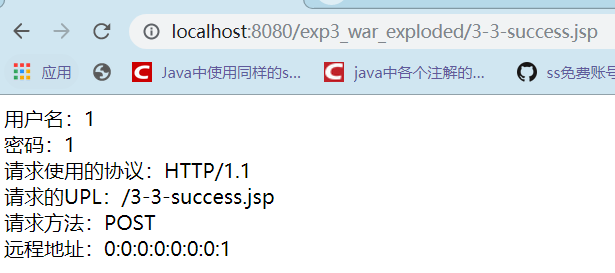


2. 编写一个JSP页面，将用户名和密码存放到会话中(假设用户名为“孤独求败”，密码为“123456”)，再重新定向到另一个JSP页面，将会话中存放的用户名和密码显示出来。(提示：使用response对象的sendRedirect()方法进行重定向。)





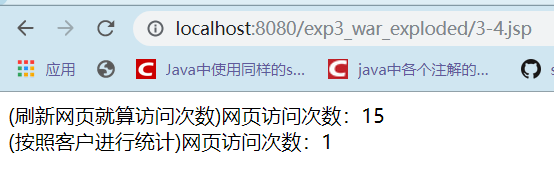
3. 编写一个JSP登录页面，可输入用户名和密码，提交请求到另一个JSP页面，该JSP页面获取请求的相关数据并显示出来。请求的相关数据包括用户输入的请求数据和请求本身的一些信息(比如请求使用的协议getProtocol() 、请求的URI request.getServletPath() 、请求方法request.getMethod() 、远程地址request.getRemoteAddr()等)。

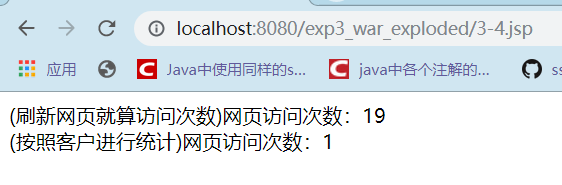
 

4. 利用隐式对象为某一网站编写一个JSP程序，统计该网站的访问次数。

一种情况是按照客户进行统计（按照浏览器进行统计，一个浏览器如果访问网站的话，就算一次访问，换句话说如果这个浏览器刷新多次网站的话，也算是一次访问）；

另一种情况是刷新一次页面，就算是一次访问。 要求用隐式对象去实现。





1. 页面跳转的两种方式(转发和重定向)区别及应用场景分析

<https://blog.csdn.net/liubin5620/article/details/79922692>

答：

1、RequestDispatcher.forward方法只能将请求转发给同一个WEB应用中的组件；而HttpServletResponse.sendRedirect 方法不仅可以重定向到当前应用程序中的其他资源，还可以重定向到同一个站点上的其他应用程序中的资源，甚至是使用绝对URL重定向到其他站点的资源。如果传递给HttpServletResponse.sendRedirect 方法的相对URL以“/”开头，它是相对于整个WEB站点的根目录；如果创建RequestDispatcher对象时指定的相对URL以“/”开头，它是相对于当前WEB应用程序的根目录。

 2、调用HttpServletResponse.sendRedirect方法重定向的访问过程结束后，浏览器地址栏中显示的URL会发生改变，由初始的URL地址变成重定向的目标URL；而调用RequestDispatcher.forward 方法的请求转发过程结束后，浏览器地址栏保持初始的URL地址不变。

 3、HttpServletResponse.sendRedirect方法对浏览器的请求直接作出响应，响应的结果就是告诉浏览器去重新发出对另外一个URL的 访问请求，这个过程好比有个绰号叫“浏览器”的人写信找张三借钱，张三回信说没有钱，让“浏览器”去找李四借，并将李四现在的通信地址告诉给了“浏览器”。于是，“浏览器”又按张三提供通信地址给李四写信借钱，李四收到信后就把钱汇给了“浏览器”。可见，“浏览器”一共发出了两封信和收到了两次回复， “浏览器”也知道他借到的钱出自李四之手。RequestDispatcher.forward方法在服务器端内部将请求转发给另外一个资源，浏览器只知道发出了请求并得到了响应结果，并不知道在服务器程序内部发生了转发行为。这个过程好比绰号叫“浏览器”的人写信找张三借钱，张三没有钱，于是张三找李四借了一些钱，甚至还可以加上自己的一些钱，然后再将这些钱汇给了“浏览器”。可见，“浏览器”只发 出了一封信和收到了一次回复，他只知道从张三那里借到了钱，并不知道有一部分钱出自李四之手。

 4、RequestDispatcher.forward方法的调用者与被调用者之间共享相同的request对象和response对象，它们属于同一个访问请求和响应过程；而HttpServletResponse.sendRedirect方法调用者与被调用者使用各自的request对象和response对象，它们属于两个独立的访问请求和响应过程。对于同一个WEB应用程序的内部资源之间的跳转，特别是跳转之前要对请求进行一些前期预处理，并要使用HttpServletRequest.setAttribute方法传递预处理结果，那就应该使用RequestDispatcher.forward方法。不同WEB应用程序之间的重定向，特别是要重定向到另外一个WEB站点上的资源的情况，都应该使用HttpServletResponse.sendRedirect方法。

5、无论是RequestDispatcher.forward方法，还是HttpServletResponse.sendRedirect方法，在调用它们之前，都不能有内容已经被实际输出到了客户端。如果缓冲区中已经有了一些内容，这些内容将被从缓冲区中。

转发和重定向的应用场景：

转发是要比重定向快，因为重定向需要经过客户端，而转发没有。有时候，采用重定向会更好，若需要重定向到另外一个外部网站，则无法使用转发。另外，重定向还有一个应用场景：避免在用户重新加载页面时两次调用相同的动作。

例如，当提交产品表单的时候，执行保存的方法将会被调用，并执行相应的动作；这在一个真实的应用程序中，很有可能将表单中的所有产品信息加入到数据库中。但是如果在提交表单后，重新加载页面，执行保存的方法就很有可能再次被调用。同样的产品信息就将可能再次被添加，为了避免这种情况，提交表单后，你可以将用户重定向到一个不同的页面，这样的话，这个网页任意重新加载都没有副作用；

但是，使用重定向不太方便的地方是，使用它无法将值轻松地传递给目标页面。而采用转发，则可以简单地将属性添加到Model,使得目标视图可以轻松访问。由于重定向经过客户端，所以Model中的一切都会在重定向时丢失。但幸运的是，在Spring3.1版本以后，我们可以通过Flash属性，解决重定向时传值丢失的问题。