**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。

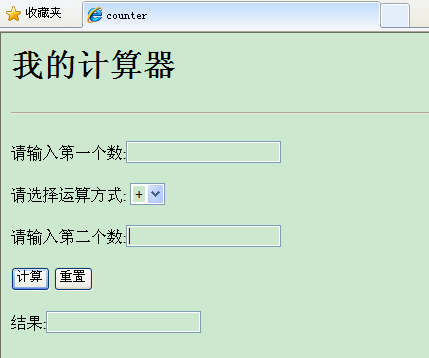


图4-24 简易计算器

**先建一个calculator.jsp页面，把参数传进去：**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>calculator.jsp</title>

</head>

<body>

<hr>

计算结果是:

<jsp:getProperty property=*"firstnum"* name=*"calcBean"* />

<jsp:getProperty property=*"operator"* name=*"calcBean"* />

<jsp:getProperty property=*"secondnum"* name=*"calcBean"* />

=<jsp:getProperty property=*"result"* name=*"calcBean"* />

<hr>

<form action=*"/exp3/WebContent/calculator.jsp"* method=*"post"*>

<table style="text-align: *center*;">

<tr>

<td colspan=*"2"*>简单的计算器</td>

</tr>

<tr>

<td>第一个参数</td>

<td><input type=*"text"* name=*"firstnum"* /></td>

</tr>

<tr>

<td><select name=*"operator"*>

<option value=*"+"*>+</option>

<option value=*"-"*>-</option>

<option value=*"\*"*>\*</option>

<option value=*"/"*>/</option>

</select></td>

</tr>

<tr>

<td>第二个数</td>

<td><input type=*"text"* name=*"secondnum"* /></td>

</tr>

<tr>

<td>

<button name=*"calc"* type=*"submit"*>计算</button>

</td>

</tr>

</table>

</form>

</body>

</html>

**再建一个index.jsp页面，处理输入的结果，进行计算，显示主界面：**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>index.jsp</title>

<script>

**function** getresult()

{

**var** num1=document.getElementById("num1").value;

**var** sig=document.getElementById("sig").value;

**var** num2=document.getElementById("num2").value;

n1 = parseFloat(num1);

n2 = parseFloat(num2);

**if**(sig=="/"&& num2=="0")

{

document.write("除数不能为零!");

}

**if**(num1=="" || num2==""){

document.write("参与计算的数值不能为空!");

}

//判断输入值是不是数字

**if**(isNaN(num1)){

document.write("第一个数输入的不是数字!");

}

**if**(isNaN(num2)){

document.write("第二个数输入的不是数字!");

}

**else**{

**switch** (sig) {

**case** "+":

document.getElementById("result").value = n1+n2;

**break**;

**case** "-":

document.getElementById("result").value =n1-n2;

**break**;

**case** "\*":

document.getElementById("result").value =n1\*n2;

**break**;

**case** "/":

document.getElementById("result").value =n1/n2;

**break**;

}

}

}

</script>

</head>

<body>

<h1>我的计算器</h1>

<hr color=*"red"*>

请输入第一个数：<input type=*"text"* name=*"num1"* id=*"num1"*><br><br>

请选择运算方式：

<select name=*"sig"* id=*"sig"*>

<option value=*"+"*>+</option>

<option value=*"-"*>-</option>

<option value=*"\*"*>\*</option>

<option value=*"/"*>/</option>

</select><br><br>

请输入第二个数：<input type=*"text"* name=*"num2"* id=*"num2"*><br><br>

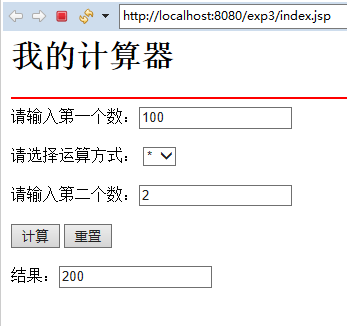
<input type=*"submit"* value=*"计算"* onclick="getresult()">

<input type=*"reset"* value=*"重置"*><br><br>

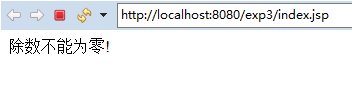
结果：<input type=*"text"* name=*"结果"* id=*"result"*/>

</body>

</html>

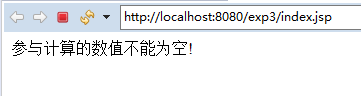


**当除数为0时，点击计算跳转页面：**



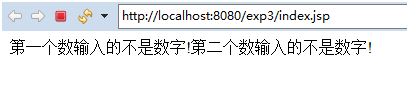
**当输入数字为空时，点击计算跳转页面：**





**当输入的不是数字时，点击计算：**





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。)

**新建一个session.jsp页面，用来放置用户名和密码的会话**：

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>会话窗口</title>

</head>

<body>

<%

HttpSession ses=request.getSession();

ses.setAttribute("username","独孤求败");

ses.setAttribute("password","123456");

response.sendRedirect("response.jsp");

%>

</body>

</html>

**再建一个response.jsp页面，显示上一个页面重定向过来的结果，把用户名和密码显示出来 ：**

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<html>

<head>

<title>response.jsp</title>

</head>

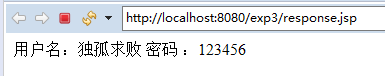
<body>

用户名：<%=request.getSession().getAttribute("username") %>

密码 ：<%=request.getSession().getAttribute("password") %>

</body>

</html>



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)。

**先新建一个login.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：**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>login.jsp</title>

</head>

<body>

<form action=*"loginSuccess.jsp"*>

用户名:<input type=*"text"* name=*"userName"*><br><br>

密码:&nbsp;&nbsp;&nbsp;&nbsp;<input type=*"text"* name=*"password"*><br><br>

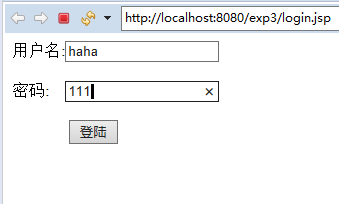
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;<input type=*"submit"* value=*"登陆"*>

</form>

</body>

</html>



**再建一个获取请求的loginSuccess.jsp页面，该JSP页面获取请求的相关数据并显示出来：**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>loginSuccess.jsp</title>

</head>

<body>

<h2>我是登陆成功后的跳转页面</h2>

<p> 用户名：<%=request.getParameter("userName") %><br><br></p>

<p>密码：<%=request.getParameter("password") %><br><br></p>

<p>请求使用的协议：<%=request.getProtocol() %><br><br></p>

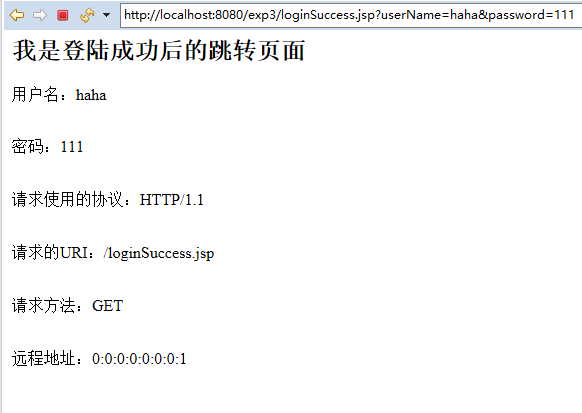
<p>请求的URI：<%=request.getServletPath() %><br><br></p>

<p>请求方法：<%=request.getMethod() %><br><br></p>

<p>远程地址：<%=request.getRemoteAddr()%><br><br></p>

</body>

</html>



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。

**新建一个count.jsp页面：**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>统计访问次数</title>

</head>

<body>

<%

**if** (pageContext.getAttribute("pageCount") == **null**) {

pageContext.setAttribute("pageCount", **new** Integer(0));

}

**if** (session.getAttribute("sessionCount") == **null**) {

session.setAttribute("sessionCount", **new** Integer(0));

}

Integer count = (Integer) pageContext.getAttribute("pageCount");

pageContext.setAttribute("pageCount", **new** Integer(

count.intValue() + 1));

Integer count2 = (Integer) session.getAttribute("sessionCount");

session.setAttribute("sessionCount", **new** Integer(

count2.intValue() + 1));

%>

页面计数：<%=pageContext.getAttribute("pageCount")%>

<br />

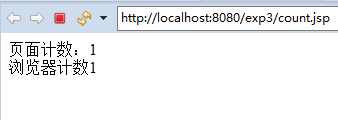
浏览器计数<%=session.getAttribute("sessionCount")%>

<br />

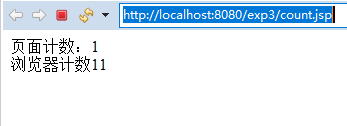
</body>

</html>

**访问一次：**



**刷新也算是访问：**



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：转发与重定向的区别：

## 两种跳转获得对象的方式

//获得转发对象getRequestDispatcher()

HttpServletRequest(httpServletRequest).getRequestDispatcher

ServletContext.getRequestDispatcher();

//获得重定向对象sendRedirect()

HttpServletResponse(httpServletResponse).sendRedirect();

## 转发和跳转的小结

1、转发使用的是getRequestDispatcher()方法;重定向使用的是sendRedirect();

2、转发：浏览器URL的地址栏不变。重定向：浏览器URL的地址栏改变；

3、转发是服务器行为，重定向是客户端行为；

4、转发是浏览器只做了一次访问请求。重定向是浏览器做了至少两次的访问请求；

5、转发2次跳转之间传输的信息不会丢失，重定向2次跳转之间传输的信息会丢失（request范围）。

## 转发和重定向的选择

1、重定向的速度比转发慢，因为浏览器还得发出一个新的请求，如果在使用转发和重定向都无所谓的时候建议使用转发。

2、因为转发只能访问当前WEB的应用程序，所以不同WEB应用程序之间的访问，特别是要访问到另外一个WEB站点上的资源的情况，这个时候就只能使用重定向了。

## 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。

要使用Flash属性，必须在Spring MVC的配置文件中添加一个<annotation-driven/>。然后，还必须再方法上添加一个新的参数类型：org.springframework.web.servlet.mvc.support.RedirectAttributes。

如下所示：

@RequestMapping(value="saveProduct",method=RequestMethod.POST)

public String saveProduct(ProductForm productForm,RedirectAttributes redirectAttributes){

//执行产品保存的业务逻辑等

//传递参数

redirectAttributes.addFlashAttribute("message","The product is saved successfully");

//执行重定向

return "redirect:/……";