六、使用 Cookies

Cookies 是重要的服务器状态保持策略。Web 服务器常使用 Cookies 技术来实现用户免登录功能和存储用户状态信息。ASIHTTPRequest 支持客户端 Cookies 的存取。

1、服务器端

Session 是服务器端技术，虽然 Cookies 是保存在客户端的。因此我们需要一个服务器端环境。打开 Eclipse，新建 Web 工程，随便写几个简单的 jsp 页面：

```
<%@ page contentType = "text/html; charset=GBK" %>
< html >
< head >
<%
String lastUrl=(String)request.getParameter( "lastUrl" );
%>
< title > login.jsp </ title >
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=iso -8859-1"-->
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=gb2312"-->
</ head >
< FORM name = "form1" METHOD = "POST" ACTION = "LoginServlet" >
< p >< input name = "username" type = "text" value = "" >< br >
< p >< input name = "pass" type = "text" value = "" >  < br >
< p >< input name = "ok" type = "submit" value = "  提交 " >
< p >< input name = "lastUrl" type = "text" value = " <%= lastUrl %>
" >
</ form >
</ html >
```

这是login.jsp，仅仅有一个用户登录表单。因为在 Web 中，往往只有经过合法登录的用户才需要保持状态。

```
<%@ page contentType = "text/html; charset=GBK" %>
< html >
< head >
<%
String uid=(String )session.getAttribute( "uid" );
String sessionid=(String)session.getId();// 获取 sessionid
String lastUrl= "index.jsp" ;
if (uid== null ){
%>
< jsp:forward page = "login.jsp" >
< jsp:param name = "lastUrl" value = "index.jsp" />
</ jsp:forward >
<% } %>
< title > index.jsp </ title >
```

```
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=iso -8859-1"-->
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=gb2312"-->
</ head >
< body >
hello <%= "," +uid+ "!"  %>
< p >
session id= <%= sessionid %>
< p >
< a href = "second_page.jsp" > goto> ></ a >
</ body >
</ html >
```

这个是index.jsp 页面，头部加入了一段 java 代码，要求用户必需登录，否则会自动转向登录页面。另外还把本页 URI 作为请求参数 lastUrl ，这样在登录成功后，会自动跳转到等路前请求的页面。

```
<%@ page contentType = "text/html; charset=GBK" language = "java" %>
< html >
< head >
<%
String uid=(String )session.getAttribute( "uid" );
String sessionid=(String)session.getId();// 获取 sessionid
if (uid== null ){
%>
< jsp:forward page = "login.jsp" >
< jsp:param name = "lastUrl" value = "second_page.jsp" />
</ jsp:forward >
<% } %>
< title > second page </ title >
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=iso -8859-1"-->
<!--meta http -equiv ="Content-Type" content="text/html ; charset
=gb2312"-->
</ head >
< body >
seconde_ page <%= "," +uid+ "!"  %>
< p >
session id= <%= sessionid %>
</ body >
</ html >
```

这个是second_page.jsp 页面。跟 index.jsp 的意思差不多，只不过想演示一下 sessionid 在多个页面中的传递。

```
public  class  LoginServlet extends  HttpServlet {
private  static  final  long  serialVersionUID  = 1L;
    protected  void  doGet(HttpServletRequest request,
```

```
HttpServletResponse response)  throws  ServletException, IOException
{
String username=request.getParameter( "username" );
String pass=request.getParameter( "pass" );
String lastUrl=request.getParameter( "lastUrl" );
if ( "" .equals(username))
username= null ;
if ( "" .equals(pass))
pass= null ;
boolean  loginSuccess= false ;
PrintWriter out=response.getWriter();
if (username!= null  && pass!= null ){
loginSuccess= true ;
}
if (loginSuccess){
// 获取session ，如果request 中session id 不存在，则新建
HttpSession session=request.getSession();
// 不设置MaxInactiveInterval ，则浏览器一关闭就过期
//   session.setMaxInactiveInterval(12*60*60);
session.setAttribute( "uid" ,username);
// 重写url ，带上 session id

String redirectUrl;
if (lastUrl!= null  && ! "" .equals(lastUrl)){
redirectUrl=response.encodeURL(lastUrl);
} else {
redirectUrl=response.encodeURL( "index.jsp" );
}
System. out .println( "redirectUrl :" +redirectUrl);
// 用forward ，不要用redirect 。因为后者是服务端转发，没有客户端请求，不会发送
cookie
RequestDispatcher rd=request.getRequestDispatcher(redirectUrl);
rd.forward(request, response);
} else {
// 获取session ，但request 中session id 不存在时，不新建
HttpSession session=request.getSession( false );
if (session!= null )
session.invalidate(); // 使session 失效
out.println( "login failed ! " );
}
}
protected  void  doPost(HttpServletRequest request,
HttpServletResponse response) throws  ServletException, IOException {
doGet(request,response);
}
```

这是servlet，用于用户登录。这里我偷了个懒，只要用户名不为空，我们就通过验证，仅仅为了演示。需要注意的就是两点：

ASIHTTPRequest系列(四)：Cookies

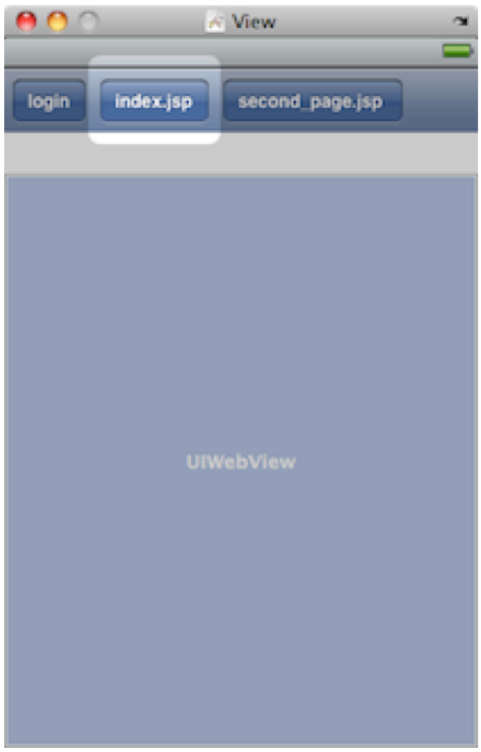
¥ 1、response.encodeURL()，这个方法服务器会检测客户端是否支持接收Cookies，如果客户端（比如IE）禁用了Cookies，则服务器会通过重写URL 的方式向客户端发送sessionid。比如，客户端请求 index.jsp 页面，encodeURL()之后服务器会把 URL 地址改写为： index.jsp;jsessionid=FC076B0E1F0763CFE7BFAFEBA2E0287C

¥ 2、页面转发。页面跳转有多种方式。比如 request.sendRedirect。在servlet中，千万不要使用 sendRedirect ，这样会使用服务器跳转，servlet 会把原来的 request 参数和 attributes 全部抛弃（包括 Cookies）。所以sendRedirect 之前和之后的请求使用的不是同一个sessionid。因此在servlet 中，我们采用的是RequestDispatcher.forward()方法，它是客户端跳转，即服务器会让客户端（浏览器）重新请求另一个地址来转发，保证 session 状态不被丢失。

进行测试，打开浏览器，访问 http://localhost:8080/test/second_page.jsp ，页面会自动跳到login.jsp。因为 java 脚本检测不到用户的 session 信息（未进行登录）。当你登录后，浏览器返回你原先请求的页面second_page.jsp。

2、iPhone 客户端

ViewController的界面很简单，1个UIWebView，1个UIToolBar，3个UIBarButtonItem：



接下来是ViewController 的 interface 代码：

```
#import <UIKit/UIKit.h>
#import "ASIHTTPRequest.h"
#define URL @
"http://220.163.103.23/interface/GetSmsList?Accounts=sa&Password=ydtf@
```

武汉大学

```
@interface CokieSessionViewController : UIViewController {
UIBarItem * button ,* indexButton ,* secondPageButton ;
UIWebView * webView ;
ASIHTTPRequest * request ;
NSURL * url ;
  }
@property ( retain ,nonatomic ) IBOutlet UIBarItem* button,
*indexButton,*secondPageButton;
@property ( retain ,nonatomic ) IBOutlet UIWebView* webView;
-( IBAction )login;
-( IBAction )gotoIndexPage;
-( IBAction )gotoSecondPage;
@end
```

将所有IBOutlet 和 IBAction 对象，在 IB 中建好连接。下面是implement代码：

```
@implementation CokieSessionViewController
@synthesize button,indexButton,secondPageButton,webView;
- ( void )didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super  didReceiveMemoryWarning ];
}

- ( void )viewDidUnload {
    [super  viewDidUnload ];
}


- ( void )dealloc {
    [super  dealloc ];
}
-( IBAction )login{
BOOL success;
url = [[[ NSURL  alloc ] initWithString :
@"http://localhost:8080/test/LoginServlet?username=dd&pass=1" ]
autorelease ];
request = [[[ ASIHTTPRequest  alloc ] initWithURL : url ] autorelease
];
// 设置 cookie 使用策略：使用（默认）
[ request  setUseCookiePersistence : YES ];
[ request  startSynchronous ];
NSString * html=[ request  responseString ];
NSRange range=[html  rangeOfString : @"login failed ! "  options :
NSCaseInsensitiveSearch ];
if (range. location == NSNotFound ) { // 如果登录成功
[ webView  loadHTMLString : @"login success"  baseURL : url ];
} else { // 如果登录失败
[ webView  loadHTMLString : @"login failed"  baseURL : url ];
```

```
}

// 获得本地cookies 集合（在第一次请求时服务器已返回 cookies，
//   虽然其中很可能只有一个cookie: sessionid ）
NSArray *cookies = [ request  responseCookies ];
// 打印sessionid
NSHTTPCookie *cookie = nil ;
for (cookie in cookies) {
if ([[cookie name ] isEqualToString : @"JSESSIONID" ]) {
NSLog ( @"session name:%@,value:%@" ,[cookie name ],[cookie value ]);
}
}
}
-( IBAction )gotoIndexPage{
url = [[[ NSURL  alloc ] initWithString :
@"http://localhost:8080/test/index.jsp" ] autorelease ];
request = [[[ ASIHTTPRequest  alloc ] initWithURL : url ] autorelease
];
// 设置 cookie 使用策略：使用（默认）
[ request  setUseCookiePersistence : YES ];
[ request  startSynchronous ];
[ webView  loadHTMLString :[ request  responseString ] baseURL : url
];
}
-( IBAction )gotoSecondPage{
url = [[[ NSURL  alloc ] initWithString :
@"http://localhost:8080/test/second_page.jsp" ] autorelease ];
request = [[[ ASIHTTPRequest  alloc ] initWithURL : url ] autorelease
];
// 设置 cookie 使用策略：使用（默认）
[ request  setUseCookiePersistence : YES ];
[ request  startSynchronous ];
[ webView  loadHTMLString :[ request  responseString ] baseURL : url
];
}
@end
```

编译运行，你可以看到，当点击 index.jsp 和 second_page.jsp 按钮时，webView 中显示的始终是登录界面 login.jsp。因为我们没有获得合法的 session。当然，第一次请求始终可以获得一个 sessionid，然而和登录后的sessionid 不一样，这个 sessionid中不保存任何用户信息。

而点击login 按钮后，再来点击那两个按钮，则可以显示相应的页面。因为服务器已经在那个sessionid 所对应的session 中放入了用户的ID。我们只需在对应的 jsp 页面中检测用户ID 就可知道用户是否已登录。