## SSO简介

SSO 一次登录 处处穿梭

比如校园卡

比如新浪微博 新浪博客，只要登录一次，就可以不用重复登录

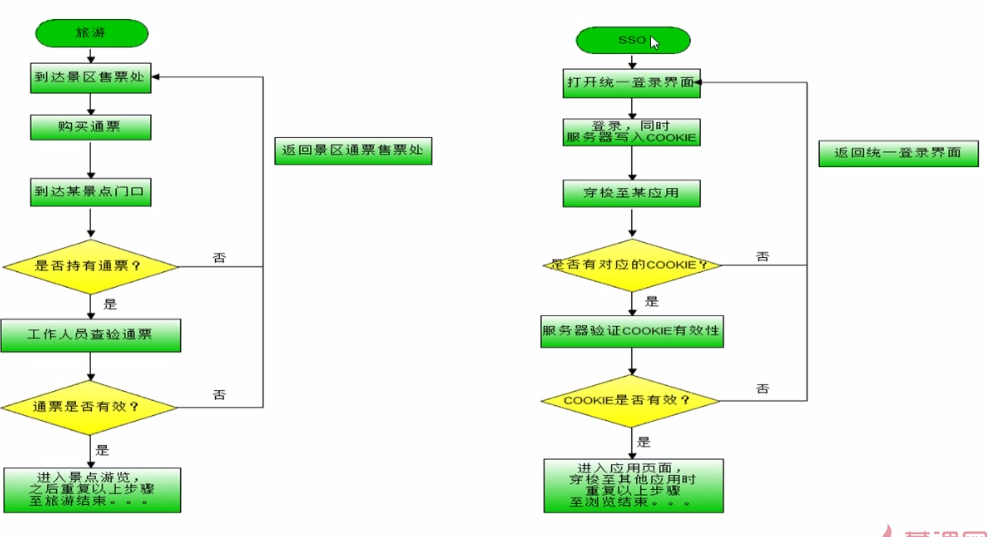
SSO Single Sign on，单点登录，保证一个账户在多个系统上实现单一用户的登录

## SSO分类

同域SSO：web1.x.com web2.x.com web3.x.com 都在x.com域名下面

跨域SSO：微博和新浪位于不同域下面 weibo.com sina.com

## SSO实现步骤和原理



### 同域SSO



用户登录之后，需要对用户名和密码进行校验，校验成功之后，如果合法，则需要在服务端创建cookie，并将cookie作用域设置为/，这样demo1和demo2都可以访问到该cookie。

（这里二级域名相同，都是localhost）

一级域名:abc.com。由一个合法的字符串+域名后缀组成（.com,.net等）

二级域名:www.abc.com，这里的www其实是主机名。二级域名其实就是一级域名下的主机名。

注意需要在登录表单中有一个隐藏表单，作为登录成功之后，需要转向的是demo1页面还是demo2页面

#### 代码

##### 工具类SSOCheck

public class SSOCheck {

//验证是否登录

public static boolean checklogin(String username,String password){

if(username.equals("123")&& password.equals("123")){

return true;

}

return false;

}

//这里校验cookie是否有效（应在拦截器中设置）

public static boolean checkcookie(HttpServletRequest request){

Cookie[] cookies = request.getCookies();

if(cookies!=null){

for(Cookie cookie:cookies){

if(cookie.getName().equals("ssocookie")&& cookie.getValue().equals("sso")){

return true;

}

}

}

return false;

}

}

##### SSOAction

这是一个统一的登录接口

/\*\*

\* 登录校验

\* 1、校验用户名密码

\* 2、校验cookie

\* @author xuzj

\*

\*/

public class SSOAction extends ActionSupport{

private String username;

private String password;

private String gotourl;

public String dologin(){

//同域下的sso

boolean ok = SSOCheck.checklogin(username, password);

if(ok){

Cookie cookie = new Cookie("ssocookie", "sso");

cookie.setPath("/");//设置到顶层

HttpServletResponse response = ServletActionContext.getResponse();

response.addCookie(cookie);

return SUCCESS;

}

return "login";

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getGotourl() {

return gotourl;

}

public void setGotourl(String gotourl) {

this.gotourl = gotourl;

}

}

##### Demo1Action

不同域下的主页demo1和demo2

//先校验cookie的有效性 在校验登录

public class Demo1Action extends ActionSupport{

private String gotourl;

public String main(){

HttpServletRequest request = ServletActionContext.getRequest();

boolean ok = SSOCheck.checkcookie(request);

if(ok){

return SUCCESS;

}

gotourl = "/demo1/main.action";

return "login";

}

public String getGotourl() {

return gotourl;

}

public void setGotourl(String gotourl) {

this.gotourl = gotourl;

}

}

##### Demo2Action

public class Demo2Action extends ActionSupport{

private String gotourl;

public String main(){

HttpServletRequest request = ServletActionContext.getRequest();

boolean ok = SSOCheck.checkcookie(request);

if(ok){

return SUCCESS;

}

gotourl = "/demo2/main.action";

return "login";

}

public String getGotourl() {

return gotourl;

}

public void setGotourl(String gotourl) {

this.gotourl = gotourl;

}

}

##### Struts.xml



在springmvc中可以使用requestMapping标注在类上，这样就起到了命名空间的作用。

@RequestMapping("/springmvc")

@Controller

public class SpringMVCTest {

private static final String SUCCESS = "success";

@RequestMapping("/testRequestMapping")

public String testRequestMapping()

{

System.out.println("testRequestMapping");

return SUCCESS;

}

}

这个方法相对于web应用的路径是：/springmvc/testRequestMapping   
若web应用的路径是：http://localhost:8080/springmvc   
则，该请求的完整路径是：   
http://localhost:8080/springmvc/springmvc/testRequestMapping

##### Login.jsp

<%@ 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>登录</title>

</head>

<body>

<center>

<h1>请登录</h1>

<form action="/sso/dologin.action" method="POST">

<span>用户名：</span>

<input type="text" name="username"/>

<span>密码：</span>

<input type="password" name="password">

<input type="text" name="gotourl" value="${gotourl }"/>

<input type="submit" />

</form>

</center>

</body>

</html>

##### success1.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!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=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

欢迎访问demo1

</body>

</html>

##### success2.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!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=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

欢迎访问demo2

</body>

</html>

总结：访问demo1.action时，首先校验cookie有效，无效，会跳转到login.jsp，访问demo2.action时，会跳转到login.jsp。此时在demo1跳转的login.jsp进行登录，登录成功之后，设置cookie，此时再去访问demo2.action时，会校验cookie有效，如果有效则不需要登录，直接访问。

### 父域相同，子域不同（主机名不一样）



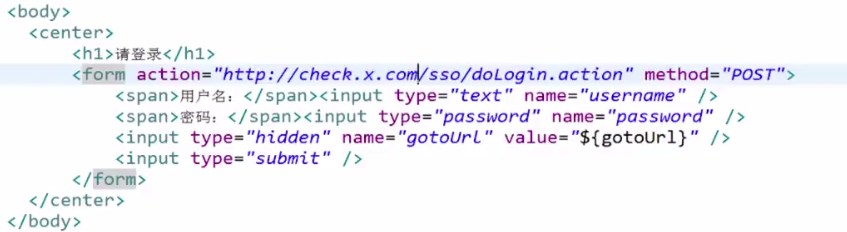
修改hosts文件，模拟不同的域名





#### 统一登录接口

##### Login.jsp



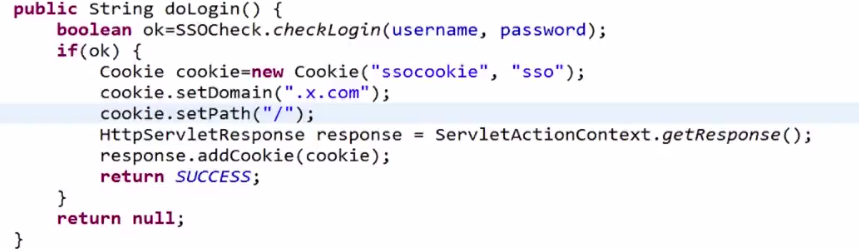
这里需要改成绝对路径

##### SSOAction

按道理应该开启3个tomcat，分别对应3个域名。

这里使用三个包模拟三个不同的服务器。

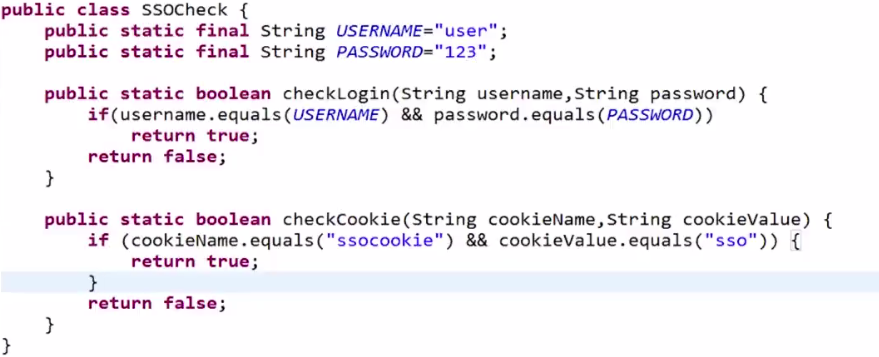
Check.x.com demo1.x.com demo2.x.com



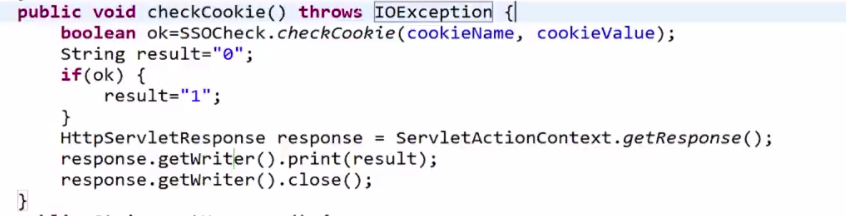
这里设置cookie的域为父域，然后设置/表示在父域的顶层，父域下的所有应用都可以访问到该cookie。如果不设置域，则代表的是当前域。

#### 登录校验接口

##### SSOCheck



##### SSOAction



通过属性驱动注入cookieName和cookieValue

#### 主页编写

##### Demo1Action

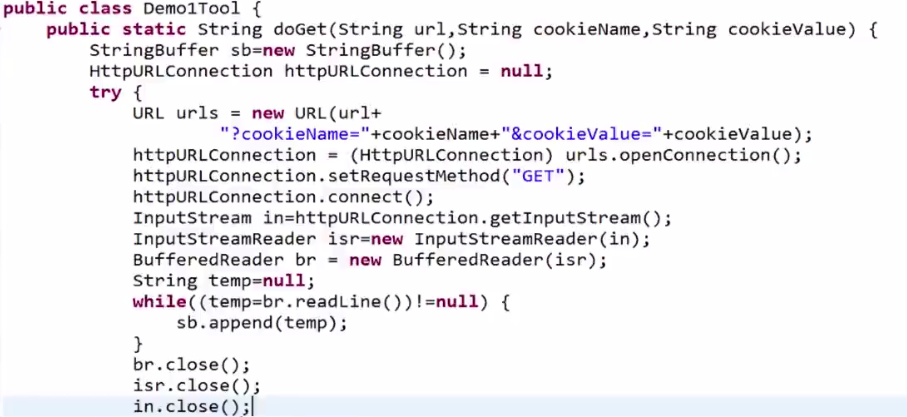


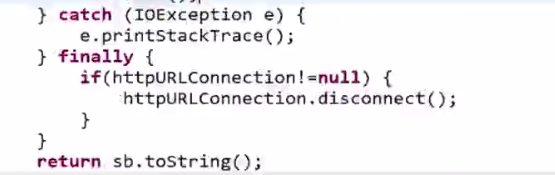
这里调用doget方法请求其他服务器上的服务。

##### Demo2Action



##### Demo1Tool





##### Demo2Tool

同demo1Tool

#### Struts.xml



### 完全跨域

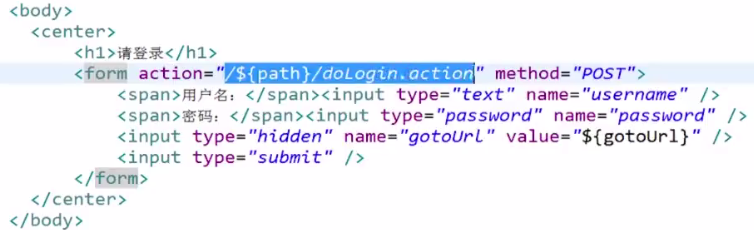




#### 统一登录接口

如果向x域提交申请，最终的cookie只能写到x域中。但是应该将cookie写到a和b域中。

在本域提交申请，但是不是在本域处理申请。申请的处理是由x域处理。



##### Demo1Action



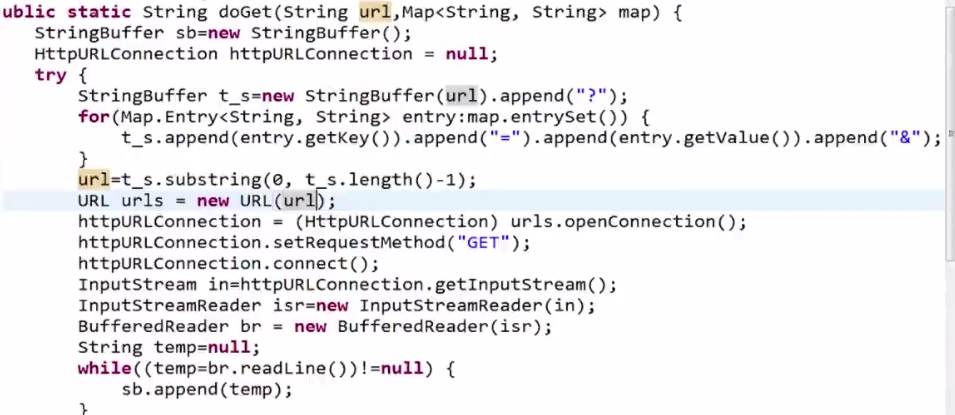
这里是在登录成功之后，会触发demo1和demo2写cookie

对应在success1.jsp中



使用iframe，让不同域之间的内容都写进来/

##### Demo1Tool

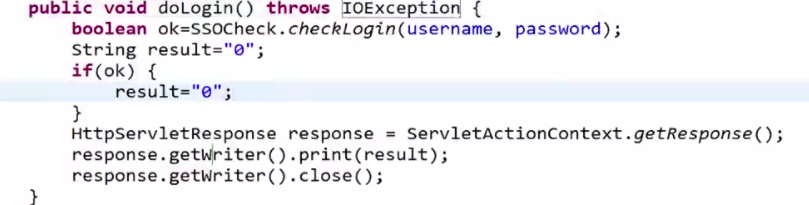


这里改为传递map参数，因为会接收用户名 密码 cookiename cookievalue，这样写更具有通用性。

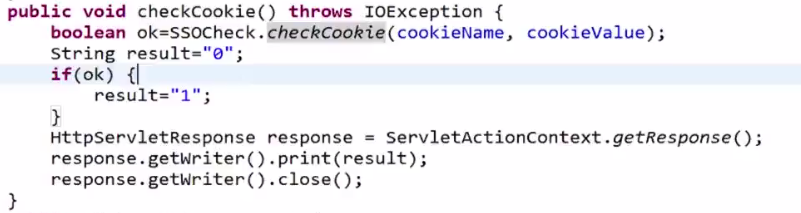
##### Demo2Tool

同demo1Tool

##### SSOAction



#### 登录校验接口



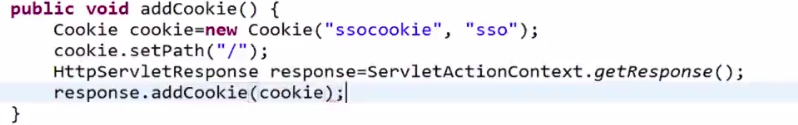
#### Demo1和demo2主页





##### 为本域设置cookie

Demo1Action和demo2Action都需要设置



#### Struts.xml



