9.2、SpringBoot-DDKJ实现登出

# 准备工作

## 1、cas服务端8443

## 2、cas客户端8083，8084 （使用DDKJ配置cas参数，二者只有端口不一样以及index.html不一样）

## 3、修改host添加域名（如果不添加域名抱歉你基本上完蛋了，如果都是localhost造成的情况有：只能登录一个客户端，如果登录另外一个客户端，则前面的会被顶掉，因为一个域名，session是一样的会被替换掉，而我们这里用的ticket，session是不一样的，如果是单点登出，那肯定是都出去了，另一个都没登上过，可不是都登出么。这个就是涉及到session问题了，一个localhost基本上就是一个session），添加了域名，但是http请求，会造成所有的客户端都必须走登录页面，而不都一个客户端登录，所有的客户端同时登录，单点登出也是各自登出各自的毫无违和感。如果是https则会解决一个登陆其余的自动登陆，但是一个登出，全部登出，我这里没有成功。哈哈哈牛逼，还是京帅哥厉害呀

# cas 单点登录

127.0.0.1 passport.sso.com

127.0.0.1 casClientOne

127.0.0.1 casClientTwo

# 客户端

## 1、客户端casconfig如下

package com.cas.healerjean.client.config;  
  
import org.jasig.cas.client.authentication.AuthenticationFilter;  
import org.jasig.cas.client.session.SingleSignOutFilter;  
import org.jasig.cas.client.session.SingleSignOutHttpSessionListener;  
import org.jasig.cas.client.util.AssertionHolder;  
import org.jasig.cas.client.util.AssertionThreadLocalFilter;  
import org.jasig.cas.client.util.HttpServletRequestWrapperFilter;  
import org.jasig.cas.client.validation.Cas30ProxyReceivingTicketValidationFilter;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.web.servlet.FilterRegistrationBean;  
import org.springframework.boot.web.servlet.ServletListenerRegistrationBean;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.web.authentication.logout.LogoutFilter;  
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;  
  
/\*\*  
 \* CasFilterConfig  
 \*  
 \* @author chuan.ma  
 \* @since 2017/6/27  
 \*/  
@Configuration  
public class CasConfig {  
  
 @Value("${cas.server.url.login}")  
 public String casServerLoginUrl;  
  
 @Value("${cas.server.url.prefix}")  
 public String casServerUrlPrefix;  
  
 @Value("${cas.client.name}")  
 public String casClientName;  
  
 /\*\*  
 \* 用于实现单点登出功能  
 \*/  
 @Bean  
 public ServletListenerRegistrationBean<SingleSignOutHttpSessionListener> singleSignOutHttpSessionListener() {  
 ServletListenerRegistrationBean<SingleSignOutHttpSessionListener> listener = new ServletListenerRegistrationBean<>();  
 listener.setEnabled(true);  
 listener.setListener(new SingleSignOutHttpSessionListener());  
 listener.setOrder(1);  
 return listener;  
 }  
  
  
 /\*\*  
 \* 该过滤器用于实现单点登出功能，单点退出配置，一定要放在其他filter之前  
 \*/  
 @Bean  
 public FilterRegistrationBean logOutFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
  
 LogoutFilter logoutFilter = new LogoutFilter(casServerUrlPrefix + "/logout?service=" + casClientName,new SecurityContextLogoutHandler());  
 filterRegistration.setFilter(logoutFilter);  
 filterRegistration.setEnabled(true);  
 filterRegistration.addUrlPatterns("/logout");  
 filterRegistration.addInitParameter("casServerUrlPrefix", casServerUrlPrefix);  
 filterRegistration.addInitParameter("serverName", casClientName);  
 filterRegistration.addInitParameter("redirectAfterValidation", "true");  
 filterRegistration.setOrder(2);  
 return filterRegistration;  
 }  
  
 /\*\*  
 \* CAS Single Sign Out Filter  
 \* 该过滤器用于实现单点登出功能，单点退出配置，一定要放在其他filter之前  
 \*/  
 @Bean  
 public FilterRegistrationBean singleSignOutFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
 filterRegistration.setFilter(new SingleSignOutFilter());  
 filterRegistration.addUrlPatterns("/\*");  
 filterRegistration.addInitParameter("casServerUrlPrefix", casServerUrlPrefix);  
 filterRegistration.addInitParameter("serverName", casClientName);  
 filterRegistration.setOrder(3);  
 return filterRegistration;  
 }  
  
 /\*\*  
 \* 该过滤器负责用户的认证工作  
 \*/  
 @Bean  
 public FilterRegistrationBean authenticationFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
 filterRegistration.setFilter(new AuthenticationFilter());  
 filterRegistration.addUrlPatterns("/\*");  
 filterRegistration.addInitParameter("ignorePattern","/error|/public\*|/assets\*|/ftl\*");  
 filterRegistration.addInitParameter("casServerLoginUrl", casServerLoginUrl);  
 filterRegistration.addInitParameter("encoding","UTF-8");  
 filterRegistration.addInitParameter("serverName", casClientName);  
 filterRegistration.addInitParameter("useSession", "true");  
 filterRegistration.setOrder(4);  
 return filterRegistration;  
 }  
  
  
 /\*\*  
 \* 该过滤器负责对Ticket的校验工作  
 \*/  
 @Bean  
 public FilterRegistrationBean cas20ProxyReceivingTicketValidationFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
 Cas30ProxyReceivingTicketValidationFilter filter = new Cas30ProxyReceivingTicketValidationFilter();  
 filterRegistration.setFilter(filter);  
 filterRegistration.addUrlPatterns("/\*");  
 filterRegistration.addInitParameter("encoding","UTF-8");  
 filterRegistration.addInitParameter("casServerUrlPrefix", casServerUrlPrefix);  
 filterRegistration.addInitParameter("serverName", casClientName);  
 filterRegistration.setOrder(5);  
 return filterRegistration;  
 }  
  
 /\*\*  
 \* 该过滤器对HttpServletRequest请求包装， 可通过HttpServletRequest的getRemoteUser()方法获得登录用户的登录名  
 \*/  
 @Bean  
 public FilterRegistrationBean httpServletRequestWrapperFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
 filterRegistration.setFilter(new HttpServletRequestWrapperFilter());  
 filterRegistration.setEnabled(true);  
 filterRegistration.addUrlPatterns("/\*");  
 filterRegistration.setOrder(6);  
 return filterRegistration;  
 }  
  
 /\*\*  
 \* 该过滤器使得可以通过org.jasig.cas.client.util.AssertionHolder来获取用户的登录名。  
 \* 比如AssertionHolder.getAssertion().getPrincipal().getName()。  
 \* 这个类把Assertion信息放在ThreadLocal变量中，这样应用程序不在web层也能够获取到当前登录信息  
 \*/  
 @Bean  
 public FilterRegistrationBean assertionThreadLocalFilter() {  
 FilterRegistrationBean filterRegistration = new FilterRegistrationBean();  
 filterRegistration.setFilter(new AssertionThreadLocalFilter());  
 filterRegistration.setEnabled(true);  
 filterRegistration.addUrlPatterns("/\*");  
 filterRegistration.setOrder(7);  
 return filterRegistration;  
 }  
  
  
 public static class RemoteUserUtil {  
  
 public static Boolean hasLogin(){  
 return AssertionHolder.getAssertion() != null;  
 }  
  
 /\*\*  
 \* 获取单点登录用户id  
 \* @return  
 \*/  
 public static Long getRemoteUserId(){  
 Object userId = AssertionHolder.getAssertion().getPrincipal().getAttributes().get("id");  
 return userId == null ? null : Long.parseLong(userId.toString());  
 }  
  
 /\*\*  
 \* 获取单点登录用户账户  
 \* @return  
 \*/  
 public static String getRemoteUserAccount(){  
 return AssertionHolder.getAssertion().getPrincipal().getName();  
 }  
  
 /\*\*  
 \* 获取单点登录用户名称  
 \* @return  
 \*/  
 public static String getRemoteUserName(){  
 Object userName = AssertionHolder.getAssertion().getPrincipal().getAttributes().get("name");  
 return userName == null ? null : userName.toString();  
 }  
  
 /\*\*  
 \* 是否超级管理员  
 \* @return  
 \*/  
 public static boolean getRemoteUserSuper(){  
 Object isSuper = AssertionHolder.getAssertion().getPrincipal().getAttributes().get("isSuper");  
 return isSuper != null && isSuper.toString().equals("1");  
 }  
 }  
}

## 2、客户端配置文件如下

server.port=8083  
  
#cas  
cas.server.url.prefix=http://passport.sso.com:8443/cas  
cas.server.url.login=${cas.server.url.prefix}/login  
cas.client.name=http://casClientOne:${server.port}

server.port=8084  
  
#cas  
cas.server.url.prefix=http://passport.sso.com:8443/cas  
cas.server.url.login=${cas.server.url.prefix}/login  
cas.client.name=http://casClientTwo:${server.port}

## 3、controller如下

@Controller  
public class HomeController {  
  
  
  
 @GetMapping("/start")  
 public String logout(){  
 return "index";  
 }  
}

## 4、thymeleaf模板如下

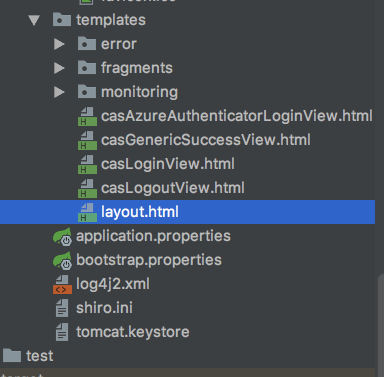
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8"/>  
 <title>Title</title>  
</head>  
<body>  
<a href="http://casClientOne:8083/logout">http://casClientOne:8083/logout</a>  
</body>  
</html>

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8"/>  
 <title>Title</title>  
</head>  
<body>  
<a href="http://casClientTwo:8084/logout">http://casClientTwo:8084/logout</a>  
</body>  
</html>

# 4、服务端

## 1、不需要配置service，jSON logoutUrl，因为默认会自动请求所有的域名。让他们全部登出。

## 2、这里用到了DDKJ提供的代码，有个地方需要改改，就是单点登出，或者是直接利用单点服务器url登录的是的时候，去的页面



<!DOCTYPE html>  
<html xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">  
<head>  
 <meta charset="UTF-8"/>  
 <meta name="viewport" content="width=device-width, initial-scale=1"/>  
  
 <title>广告平台</title>  
 <!--<script type="text/javascript" src="https://admore-cas.oss-cn-hangzhou.aliyuncs.com/assets/js/jquery/1.11.3/jquery.min.js"></script>-->  
 <script th:inline="javascript">  
 function *handleUrl*(cas\_url) {  
 return 'http://passport.sso.com:8443/cas';  
 }  
 var ***login\_url***=*handleUrl*(***window***.location.href);  
 function *goLogin*() {  
 ***window***.location.href=***login\_url***;  
 }  
 *goLogin*();  
 </script>  
</head>  
  
<body>  
  
</body>  
</html>

# 5、开始测试，启动服务端和客户端

### 浏览器打开<http://casclientone:8083/start> ，<http://casclienttwo:8084/start> 我们会发现，二者都需要登录，具体为毛，因为你他妈的用了http，不是https。单点扥出也是各自管各自的





# 6、https开始一个登陆其他不必登陆

## 1、服务端

#SSL配置  
server.ssl.enabled=false  
server.ssl.key-store=classpath:tomcat.keystore  
server.ssl.key-store-password=123456  
server.ssl.keyAlias=passport.sso.com  
#SSL配置

## 2、客户端

server.port=8084  
  
#cas  
#cas.server.url.prefix=http://passport.sso.com:8443/cas  
cas.server.url.prefix=https://passport.sso.com:8443/cas  
cas.server.url.login=${cas.server.url.prefix}/login  
cas.client.name=http://casClientTwo:${server.port}

server.port=8083  
  
#cas  
#cas.server.url.prefix=http://passport.sso.com:8443/cas  
cas.server.url.prefix=https://passport.sso.com:8443/cas  
  
cas.server.url.login=${cas.server.url.prefix}/login  
cas.client.name=http://casClientOne:${server.port}

## 3、测试成功，一个登陆，另外一个自动登陆，但是一个登出，其他登出，还他妈的真没成功，还是各自管各自的