Spring Security 4.0 + CAS4.0 实现单点登陆

作者：cbxjj
原文链接：https://ld246.com/article/1493286571182
来源网站：链滴

前言

Spring Security 4.0 + CAS4.0 整合的资料网上不多,我推荐这个,Spring Security 4.0 CAS实现单点录，然后配下来发现还是有很多问题,各种问题，各种不成功!

而且出了问题完全没有头绪

直接上配置好的文件

1. web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
   <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
         /WEB-INF/applicationContext.xml,/WEB-INF/spring-cas.xml
      </param-value>
   </context-param>
   <listener>
      <listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-cl
ss>
   </listener>
   <listener>
      <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class

   </listener>

   <!-- 单点登录注销过滤器 -->
   <filter>
      <filter-name>CAS Single Sign Out Filter</filter-name>
      <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
      <init-param>
         <param-name>casServerUrlPrefix</param-name>
         <param-value>https://cas.infozr.com:8443/cas/</param-value>
      </init-param>
   </filter>
   <filter-mapping>
      <filter-name>CAS Single Sign Out Filter</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
   <!-- spring字符编码过滤器 -->
   <filter>
      <filter-name>characterEncodingFilter</filter-name>
      <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
      <init-param>
         <param-name>encoding</param-name>
         <param-value>UTF-8</param-value>
      </init-param>
```

```
    </filter>
    <filter-mapping>
        <filter-name>characterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- springSecurity过滤器 -->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- session过滤器，检查是否有session -->
    <filter>
        <filter-name>onlineFilter</filter-name>
        <filter-class>cn.infozr.sso.cas.filter.OnlineFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>onlineFilter</filter-name>
        <url-pattern>*.do</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>onlineFilter</filter-name>
        <url-pattern>*.jsp</url-pattern>
    </filter-mapping>

    <!-- 定义首页 -->
    <welcome-file-list>
        <welcome-file>login.html</welcome-file>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

2. OnlineFilter.java

```
/**
 * 登陆验证过滤
 */
public class OnlineFilter extends HttpServlet implements Filter {
    private static final long serialVersionUID = 1L;
    private String redirect_url;

    public void doFilter(ServletRequest request, ServletResponse response,   FilterChain chain)
hrows IOException, ServletException {
        // 这里设置如果没有登陆将要转发到的页面
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
        HttpSession session = req.getSession(true);

        String requestUri = req.getRequestURI();
        String contextPath = req.getContextPath();
```

```
        String url = requestUri.substring(contextPath.length());
        // 从session里取的用户名信息
        String username = (String) session.getAttribute("sessionUsername");
        // 这里获取session，为了检查session里有没有保存用户信息，没有的话回转发到登陆页面
        // 判断如果没有取到用户信息,就跳转到登陆页面
        if(username == null && StringUtils.isEmpty(req.getParameter("ticket"))) {
            // 跳转到登陆页面
            if(isAjaxRequest(req)){
                try {
                    response.getWriter().print("login");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            else{
                res.sendRedirect(redirect_url);
            }

        } else {
            chain.doFilter(request, response);
        }
    }
     private boolean isAjaxRequest(HttpServletRequest request) {
        String header = request.getHeader("X-Requested-With");
        if (header != null && "XMLHttpRequest".equals(header))
            return true;
        else
            return false;
    }
    public void init(FilterConfig filterConfig) throws ServletException {
        StringBuilder url = new StringBuilder();
        url.append("https://cas.infozr.com:8443/cas");
        url.append("/login?service=");
        url.append("http://192.168.0.110:8080/infozr_upms/login/cas");
        redirect_url = url.toString();
    }
}
```

3. spring-cas.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:sec="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-4.2.xsd
       http://www.springframework.org/schema/security
       http://www.springframework.org/schema/security/spring-security-4.2.xsd"
       default-lazy-init="true">
```

```
    <!-- 浏览权限设定，根据自己的情况修改 -->
    <sec:http auto-config="false" use-expressions="false" disable-url-rewriting="false"
            entry-point-ref="casProcessingFilterEntryPoint">
        <sec:headers disabled="true"/>
        <sec:intercept-url pattern="/login/cas*" access="IS_AUTHENTICATED_ANONYMOUSLY"
>
        <sec:intercept-url pattern="/**" access="IS_AUTHENTICATED_ANONYMOUSLY"/>
        <sec:custom-filter position="CAS_FILTER" ref="casAuthenticationFilter" />
        <sec:logout logout-success-url="https://cas.infozr.com:8443/cas/logout"/>
        <sec:custom-filter ref="requestSingleLogoutFilter" before="LOGOUT_FILTER"/>
        <sec:custom-filter ref="singleLogoutFilter" before="CAS_FILTER"/>
        <sec:csrf disabled="true"/>
        <!--<sec:custom-filter ref="csrfTokenFilter" after="CSRF_FILTER"/>-->
    </sec:http>

    <!-- CSRF Configuration
    <bean id="csrfTokenFilter" class="com.test.cloud.security.CsrfTokenFilter"/>
    -->

    <sec:authentication-manager alias="authenticationManager">
        <sec:authentication-provider ref="casAuthenticationProvider"/>
    </sec:authentication-manager>


    <!-- 注销客户端 -->
    <bean id="singleLogoutFilter" class="org.jasig.cas.client.session.SingleSignOutFilter"/>
    <!-- 注销服务器端 -->
    <!-- This filter redirects to the CAS Server to signal Single Logout should be performed -->

    <bean id="requestSingleLogoutFilter" class="org.springframework.security.web.authenticat
on.logout.LogoutFilter">
        <constructor-arg value="http://cas.infozr.com:8080/cas/logout"/>
        <constructor-arg>
            <bean class="org.springframework.security.web.authentication.logout.SecurityContex
LogoutHandler"/>
        </constructor-arg>
        <property name="filterProcessesUrl" value="/logout/cas"/>
    </bean>



    <bean id="casAuthenticationFilter" class="org.springframework.security.cas.web.CasAuthe
ticationFilter">
        <property name="authenticationManager" ref="authenticationManager"/>
        <!-- 认证成功返回的页面，此处做了修改，这个类是继续之前的操作。默认的类是设置一个固
的页面  -->
        <!-- 登录成功处理，将用户信息存入session -->
        <property name="authenticationSuccessHandler">
            <bean class="cn.infozr.sso.cas.service.CasAuthenticationSuccessHandler">
                <!-- cas登录成功后跳转页面 -->
                <property name="defaultTargetUrl" value="/admin.jsp"/>
                <!-- <property name="usersDao" ref="resourcesDao"/>
                <property name="resDao" ref="resourcesDao"/>  -->
            </bean>
```

```
        </property>
        <!-- 认证失败返回的页面(非403错误)  -->
        <property name="authenticationFailureHandler">
            <bean class="org.springframework.security.web.authentication.SimpleUrlAuthenticati
nFailureHandler">
                <property name="defaultFailureUrl" value="/static/html/errors/403.html"/>
            </bean>
        </property>
    </bean>

    <bean id="casProcessingFilterEntryPoint" class="org.springframework.security.cas.web.Ca
AuthenticationEntryPoint">
        <!-- 单点登录服务器登录URL -->
        <property name="loginUrl" value="https://cas.infozr.com:8443/cas/login"/>
        <property name="serviceProperties" ref="serviceProperties"/>
    </bean>

    <bean id="userDetailsManager" class="cn.infozr.sso.cas.service.UserDetailsManager"/>

    <bean id="casAuthenticationProvider"
         class="org.springframework.security.cas.authentication.CasAuthenticationProvider">
        <!-- 注入获取tsp用户的service -->
        <property name="authenticationUserDetailsService">
            <bean class="org.springframework.security.core.userdetails.UserDetailsByNameServic
Wrapper">
                <constructor-arg ref="userDetailsManager" />
            </bean>
        </property>
        <property name="serviceProperties" ref="serviceProperties"/>
        <property name="ticketValidator">
            <bean class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
                <constructor-arg index="0" value="https://cas.infozr.com:8443/cas"/>
            </bean>
        </property>
        <property name="key" value="an_id_for_this_auth_provider_only"/>
    </bean>


    <bean id="serviceProperties" class="org.springframework.security.cas.ServiceProperties">
        <!--[login/cas]是Spring Security 4.0后修改的地址、跟3.X版本完全不同、请勿修改-->
        <!-- Cas Server认证成功后的跳转地址，这里要跳转到我们的Spring Security应用，之后会由C
sAuthenticationFilter处理，默认处理地址为/j_spring_cas_security_check -->
        <property name="service" value="http://192.168.0.110:8080/infozr_upms/login/cas"/>
        <property name="sendRenew" value="false"/>
    </bean>
</beans>
```

注意:serviceProperties里面的属性service必须和OnlineFilter里面的跳转url后面的service=的值保一致

4.CasAuthenticationSuccessHandler.java

```
public class CasAuthenticationSuccessHandler extends SimpleUrlAuthenticationSuccessHandl
r{
```

```
    private static Logger logger = LogManager.getLogger(CasAuthenticationSuccessHandler.cl
ss);

    @Resource
    private BaseDAO usersDao;

    @Resource
    private BaseDAO resDao;

    public void setUsersDao(BaseDAO usersDao) {
        this.usersDao = usersDao;
    }

    public void setResDao(BaseDAO resDao) {
        this.resDao = resDao;
    }

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse res
onse, Authentication authentication) throws IOException,
            ServletException {
        String username = ((UserDetails) authentication.getPrincipal()).getUsername();
        logger.info("cas验证成功,username={}",username);
        request.getSession().setAttribute("sessionUsername", username);

        super.onAuthenticationSuccess(request, response, authentication);
    }
}
```

5. UserDetailsManager.java

```
public class UserDetailsManager implements UserDetailsService {

    /**
     * 此处的参数[loginId]为CAS登录画面输入的用户名
     */
    @Override
    public UserDetails loadUserByUsername(final String loginId) throws UsernameNotFoundEx
eption {
        final Collection<GrantedAuthority> auths = new ArrayList<GrantedAuthority>();
        System.out.println(loginId);

        UserDetails userDetails = new UserDetails() {
            /**
             * Returns the authorities granted to the user. Cannot return
             * <code>null</code>.
             *
             * @return the authorities, sorted by natural key (never
             *         <code>null</code>)
             */
            @Override
            public Collection<? extends GrantedAuthority> getAuthorities() {
                SimpleGrantedAuthority authority = new SimpleGrantedAuthority("ROLE_USER");
```

```
    auths.add(authority);

    return auths;
}

/**
 * Returns the password used to authenticate the user.
 *
 * @return the password
 */
@Override
public String getPassword() {
    return "11";
}

/**
 * Returns the username used to authenticate the user. Cannot return
 * <code>null</code> .
 *
 * @return the username (never <code>null</code>)
 */
@Override
public String getUsername() {
    return loginId;
}

/**
 * Indicates whether the user's account has expired. An expired
 * account cannot be authenticated.
 *
 * @return <code>true</code> if the user's account is valid (ie
 *         non-expired), <code>false</code> if no longer valid (ie
 *         expired)
 */
@Override
public boolean isAccountNonExpired() {
    return true;
}

/**
 * Indicates whether the user is locked or unlocked. A locked user
 * cannot be authenticated.
 *
 * @return <code>true</code> if the user is not locked,
 *         <code>false</code> otherwise
 */
@Override
public boolean isAccountNonLocked() {
    return true;
}

/**
 * Indicates whether the user's credentials (password) has expired.
 * Expired credentials prevent authentication.
```

```
         *
         * @return <code>true</code> if the user's credentials are valid (ie
         *        non-expired), <code>false</code> if no longer valid (ie
         *        expired)
         */
        @Override
        public boolean isCredentialsNonExpired() {
            return true;
        }

        /**
         * Indicates whether the user is enabled or disabled. A disabled
         * user cannot be authenticated.
         *
         * @return <code>true</code> if the user is enabled,
         *        <code>false</code> otherwise
         */
        @Override
        public boolean isEnabled() {
            return true;
        }
    };

    return userDetails;
  }
}

</br>
</br>
```

手机游戏源码素材网：http://www.codegather.com