第三方单点登录 ECOLOGY 方案

目 录

1 ECOLOGY 配置

1.1 配置 WEB.XML

请在 weaver\ecology\WEB-INF\web.xml 最后面配置：

```
<servlet>
    <servlet-name>getToken</servlet-name>
    <servlet-class>weaver.weaversso.GetToken</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>getToken</servlet-name>
    <url-pattern>ssologin/getToken</url-pattern>
</servlet-mapping>
```

请在 weaver\ecology\WEB-INF\web.xml 中安全补丁包后面配置：

```
<filter>
    <filter-name>WeaverLoginFilter</filter-name>
    <filter-class>weaver.weaversso.WeaverLoginFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>WeaverLoginFilter</filter-name>
    <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```

配置说明:

名称	参数	作用	备注

getToken	appid loginid	该 Servlet 用于获取 SSO token	appid:应用标识,可自行定义(以下 1.2 配置属性文件时的 key 需要与之对应,例如: Client1); loginid: ecology 的登录名
WeaverLoginFilter	ssoToken	该过滤器用于验证 SSO token 的有效性	ssoToken: 统一认证中心生成的 token;

1.2 配置 WEAVERLOGINCLIENT.PROPERTIES

配置文件：

\weaver\ecology\WEB-INF\prop\weaverloginclient.properties

配置内容：

配置项	配置说明
IP	配置应用标识和所对应的 IP 地址 ，多个用逗号分隔。只有 ip 地址符合的才能获取到 token。

配置示例：

```
Client1=127.0.0.1,192.168.0.1
```

2 第三方单点登录 ECOLOGY 接入开发

2.1 示例代码

以下示例使用 HttpClient 先从统一认证中心获取 token，再请求访问 ecology 的一个产品版本展现页面，

示例代码如下：

```
package test;

import org.apache.http.*;
import org.apache.http.client.HttpClient;
```

```
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.aspectj.lang.annotation.Aspect;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.util.*;

/**
 * Created by crazyDream on 2017/1/10.
 * 测试例子
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath*:*spring*.xml")
@WebAppConfiguration
@Aspect
public class MyTest extends MockMvcRequestBuilders {


    /**
     * 获取 token
     */
    @Test
```

```
    public void getToken() {
        String url = "http://192.168.46.62:8808/ssologin/getToken";
        Map<String, String> params = new HashMap<>();
        params.put("appid", "demo1");
        params.put("loginid", "sw");
        String s = MyTest.doPost(url, params);
        System.out.println(s);
//获得的 token:
659C43561254716F8961A2586888BD17288178101C86582821E50E190F941A72
    }

    /**
     * 带上 token 访问 ecology 页面
     */
    @Test
    public void visitPage() {
        String token =
"659C43561254716F8961A2586888BD17288178101C86582821E50E190F941A72";
        String url = "http://192.168.46.62:8808/systeminfo/version.jsp";
        Map<String, String> params = new HashMap<>();
        params.put("ssoToken", token);
        String s = MyTest.doPost(url, params);
        System.out.println(s); // 能成功返回内容

    }



    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext wac;


    @Before
    public void setUp() throws Exception {
        mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
    }



    /**
     * post 请求(用于 key-value 格式的参数)
     * @param url
```

```
     * @param params
     * @return
     */
    public static String doPost(String url, Map params){

        BufferedReader in = null;
        try {
            // 定义 HttpClient
            HttpClient client = new DefaultHttpClient();
            // 实例化 HTTP 方法
            HttpPost request = new HttpPost();
            request.setURI(new URI(url));

            //设置参数
            List<NameValuePair> nvps = new ArrayList<NameValuePair>();
            for (Iterator iter = params.keySet().iterator(); iter.hasNext();) {
                String name = (String) iter.next();
                String value = String.valueOf(params.get(name));
                nvps.add(new BasicNameValuePair(name, value));

                //System.out.println(name +"-"+value);
            }
            request.setEntity(new UrlEncodedFormEntity(nvps, HTTP.UTF_8));

            HttpResponse response = client.execute(request);
            int code = response.getStatusLine().getStatusCode();
            if(code == 200){      //请求成功
                in = new BufferedReader(new InputStreamReader(response.getEntity()
                        .getContent(),"utf-8"));
                StringBuffer sb = new StringBuffer("");
                String line = "";
                String NL = System.getProperty("line.separator");
                while ((line = in.readLine()) != null) {
                    sb.append(line + NL);
                }

                in.close();

                return sb.toString();
            }
            else{     //
                System.out.println("状态码：" + code);
                return null;
            }
```

```
        }
        catch(Exception e){
            e.printStackTrace();

            return null;
        }
    }

    /**
     * post 请求（用于请求 json 格式的参数）
     * @param url
     * @param params
     * @return
     */
    public static String doPost(String url, String params) throws Exception {

        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httpPost = new HttpPost(url);// 创建 httpPost
        httpPost.setHeader("Accept", "application/json");
        httpPost.setHeader("Content-Type", "application/json");
        String charSet = "UTF-8";
        StringEntity entity = new StringEntity(params, charSet);
        httpPost.setEntity(entity);
        CloseableHttpResponse response = null;

        try {

            response = httpclient.execute(httpPost);
            StatusLine status = response.getStatusLine();
            int state = status.getStatusCode();
            if (state == HttpStatus.SC_OK) {
                HttpEntity responseEntity = response.getEntity();
                String jsonString = EntityUtils.toString(responseEntity);
                return jsonString;
            }
            else{
            }
        }
        finally {
            if (response != null) {
                try {
                    response.close();
                } catch (IOException e) {
                    e.printStackTrace();
```

```
                }
            }
            try {
                httpclient.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }



}
```

以上示例 visitPage 方法能通过验证并正确返回 ecology 的页面内容，如下图：

3 方案使用注意事项(重要)

3.1 注意事项

1,碰到请求被拦截跳转到 ecology 的登录页，则需要配置调整安全规则,方能确保该方案正常使用;

找到 ecology/WEB-INF/weaver_security_config.xml,调整 is-login-check 属性为 false,配置如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <!--enable or disable firewall, 1 enable, 0 disable-->
  <status>1</status>
  <!--enable or disable debug mode-->
  <sys-debug>true</sys-debug>
  <!--cache params remove time，default 480 min-->
```

```
    <intervalTime>480</intervalTime>
    <!--Thread scan time,  default30 min-->
    <scanTime>30</scanTime>
    <!--Sikp all referer xss -->
    <skip-ref>true</skip-ref>
    <!--Sikp all host check, check host header attack-->
    <skip-host>true</skip-host>
    <!--is must be execute xss filter-->
    <must-xss>true</must-xss>
    <!--xss type    0: black list, according xss-keyword-list filter-->
    <xss-type>0</xss-type>
    <!--print value info when xss filter-->
    <xss-debug>false</xss-debug>
    <!--is enable rule check-->
    <skip-rule>false</skip-rule>
    <!--enable or disable webservice security check, true|false, default is false-->
    <enable-webservice-check>false</enable-webservice-check>
    <is-login-check>false</is-login-check>
</root>
```

2,WeaverLoginFilter 里需添加 url-pattern *.html（E9 以上需要添加）;如:

```
<filter>
  <filter-name>WeaverLoginFilter</filter-name>
  <filter-class>weaver.weaversso.WeaverLoginFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>WeaverLoginFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
  <url-pattern>*.html</url-pattern>
</filter-mapping>
```

3,有 OA Nginx 集群的环境,需要在 weaverclient.properties 配置上 nginx 服务的 ip,如果访问接口时返回 ip 非法之类的,请检查/log/integration/integration.log 日志里的 requestip 输出,确认是否有配置到;

4,携带拼接 ssoToken 时,注意拼接位置,E9 的很多页面是单页面,不要拼在#路由地址后面(E9 以上需要注意),如:

http://127.0.0.1/wui/index.html#/main?ssoToken=XXX,这是错的,

需要这样拼:

http://127.0.0.1/wui/index.html?ssoToken=XXX#/main