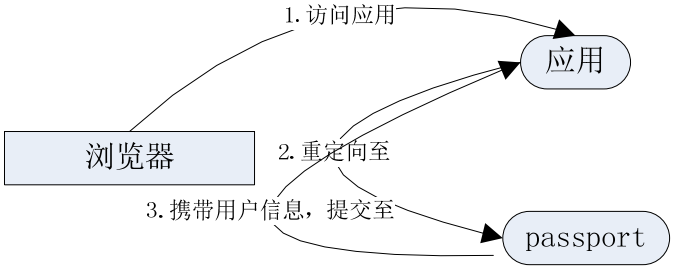
Passport 单点登录方案使用手册

说明

Passport 单点登录方案是基于 Form 表单转发的 Web 单点登录方案。该方案可以实现跨域的用户统一认证和单点登录，支持在多个站点之间互相感知用户的登录状态，支持在应用页面内嵌入登录框（IFrame 方式）。单点登录的流程如下图所示：



登录步骤：

1. 用户访问应用，应用发现用户未登录
2. 应用通过浏览器，将用户的登录界面重定向至 Passport.
3. 用户在 Passport 输入密码验证没有问题之后，将用户信息签名通过浏览器提交至应用。应用解析无误后，便完成登录过程。

接入 Passport 单点登录

准备

在正式集成 Passport 之前，需要拿到 Passport 的开发包，在开发包里包括以下 Jar 包 duckling-common-5.3.2.jar，umt-api-5.3.5.jar，xpp3_min-1.1.4c.jar， xstream-1.3.1.jar， bcprov-jdk15-1.46.jar， bcprov-jdk15-129.jar, httpclient-duckling-3.01.jar

修改 Web.xml

需要在 Web 系统中添加 LoginServlet, LogoutServlet，将下面的 XML 添加到 web.xml 中：

```
<servlet>
    <servlet-name>loginServlet</servlet-name>
    <servlet-class>cn.vlabs.duckling.api.umt.sso.LoginServlet</servlet-class>
      <init-param>
```

```
            <param-name>config</param-name>
            <param-value>WEB-INF/sso.properties</param-value>
        </init-param>
         <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>loginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>logoutServlet</servlet-name>
        <servlet-class>cn.vlabs.duckling.api.umt.sso.LogoutServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>logoutServlet</servlet-name>
        <url-pattern>/logout</url-pattern>
    </servlet-mapping>
```

这里的 login 和 logout 的 url-pattern 可以根据实际的需要进行配置。

sso.properties 配置示例：

```
umt.baseUrl = http://passport.escience.cn
umt.login.url=${umt.baseUrl}/login
umt.logout.url=${umt.baseUrl}/logout
umt.publicKey.url=${umt.baseUrl}/getUMTPublicKey
umt.login.exthandle.class=cn.vlabs.duckling.vmt.ui.servlet.VMTSSOLoginHandleImpl
umt.auth.appname=coremail

localapp.logout.url=${local.baseUrl}/logout
localapp.login.return=${local.baseUrl}
```

其中 umt.login.exthandle.class 是实现 cn.vlabs.duckling.api.umt.sso.ILoginHandle 接口的类，接口定义如下：

```
public interface ILoginHandle {
      /**
         * 用户登陆之前，系统调用此方法 <br>
         * 可以在此方法中设置一些参数传到 umt
         * @param request
         * @param response
         * @param extParamsToUmt
         */
        void initBeforLogin(HttpServletRequest request,
HttpServletResponse response,Map<String,String> extParamsToUmt);
    /**
```

```
     * 用户登录成功后，初始化 UserContext 的额外信息 <br>
     * 例如： <br>
     * 添加用户的 principals
     * @param request
     * @param response
     * @param context
     */
        void initAfterLogin(HttpServletRequest request,
HttpServletResponse response,UserContext context);
    /**
     * 用户退出之前，系统调用此方法
     * @param request
     * @param response
     * @param context
     */
        void destroyBeforeLogout(HttpServletRequest
request,HttpServletResponse response,UserContext context);

}
```

用户登录以后，会在 UserContext 中记录用户的身份，信息。

可以通过 UserContext. getUserPrincipal ()获得登录用户的身份。下面是判断是否登录成功，并获取用户身份的示例代码：

```
if (usrctx.isAuthenticated()){
  UserPrincipal principal = usrctx.getUserPrincipal();
  System.out.printf("Current user id=%s, displayName=%s\n, email=%s",
        principal.getName(), principal.getDisplayName(), principal.getEmail());
}
```

添加登录状态感知功能

在用户刚进入系统，还未进入登录状态时，在需要感知的页面引入 passport 的 JavaScript（该 JavaScript 依赖 JQuery ）。

示例代码如下：

```
<vwb:UserCheck status="notAuthenticated">
    <script type="text/javascript"
src="http://passport.escience.cn/js/passport.js"></script>
    <script type="text/javascript">
        $(document).ready(function(){
            var saveProfileUrl="${url_to_login}";
            var passport=new Passport({
                umtUrl:"http://passport.escience.cn/",
```

```
                viewPort:"#content",
                message:"<fmt:message key='login.doing'/>"
            });
            passport.checkAndLogin(saveProfileUrl,{appname:"coremail"});
        });
    </script>
</vwb:UserCheck>
```

在引入之后，需要创建 Passport 对象。下面是各个参数的含义：

umtUrl:设为 Passport 的入口地址即可，如 http://passport.escience.cn/

viewPort:显示登录中消息的时候，依附的 Dom 对象。登录时，会在页面的右上角显示正在登录的消息框，该消息框的位置依据提供的 Dom 对象计算右边坐标。

message:在登录是显示的信息。

初始化完成以后，调用 passport.checkAndLogin(saveProfileUrl,{appname:"coremail"});

saveProfileUrl 是刚才第 2 部分要求配置的 LoginServlet 的 URL 的地址。

appname 设置成 coremail 即可。

添加基于 IFrame 的登录框

添加基于 IFrame 的登录框的操作类似于添加登录感知的代码，代码示例如下所示：

```
<script type="text/javascript"
src="http://passport.escience.cn/js/passport.js"></script>
<script type="text/javascript">
$(document)
        .ready(
                function() {
                    var saveProfile = "${login_url}";
                    var passport = new Passport({
                        umtUrl : "http://passport.escience.cn/",
                        viewPort : "#header"
                    });
                    passport.showIframe("#container", saveProfile, {
                        appname : "coremail",
                        target : "_parent",
                        theme : "iframe",
                        width : "300px",
                        height : "207px"
                    });
                });
```

</script>

初始化的部分如在第 3 部分所示。
在使用基于 IFrame 的方式登录框时，调用新的方法, passport.showIframe(containerDiv, loginUrl, options);
其中，containerDiv 是容纳 IFrame 的容器，用于显示 IFrame。
loginUrl 是第一步中配置的 LoginServlet 的 url。
options 中可以自定义一些参数

appname:继续使用 coremail
target:在 Iframe 时，设为_parent
theme:在 Iframe 时，缺省设置为 iframe（可根据需要定制登录界面.)
width，height:是 Iframe 的高和宽。

总结起来，接入的工作需要：

1. 配置 LoginServlet，LogoutServlet
2. 实现 ILoginHandler，并配置 sso.properties 文件。
3. 在需要感知登录的页面，引入 passport.js，并调用 checkAndLogin 方法。
4. 在需要引入 IFrame 登录界面的地方，引入 passport.js，并调用 showIframe 方法。