Single Sign On

Introduction

With Elixir Repertoire Server 7.x onwards, all resources (*.rml, *.ds, *.job) are accessible via the REST API. All resources, at first attempt, will be prompted to enter a valid username and password. This ensure that all resources are secure and not easily accessible from the web.

SSO in Repertoire Server

Q: How can a 3rd party Web Application integrate with the Repertoire Server and access all resources automatically ?

A: This can be easily achieved using the following approach.

- Integrating Repertoire Server in Java
- Integrating Repertoire Server in .Net

Integrating Repertoire Server in Java

- Download the sample RepertoireSSO-7.x.x.war application.

- This war application contains jar libraries which are specific for Repertoire Server 7.x.x.
    - Do not attempt to use this war application with Repertoire Server 6.x and below.

- The RepertoireSSO.war is a simple URL redirect application illustrating how single-sign-on can be achieved between 2 web applications.

- The **ParamBaseRedirec**t servlet will attempt to login to the Repertoire Server using the login username/password information from the **web.xml** file
    - Upon successful authentication, the **ParamBasedRedirect** servlet will redirect to a single intended Repertoire Server URL (ie the "return" value). Typically, this should be the main home page of the application.

- Edit *RepertoireSSO\WEB-INF\web.xml* with the necessary information.

    ....

<init-param>
<param-name>url</param-name>
<param-value>http://localhost:8080/</param-value>
<= hostname/port of your Elixir Repertore Server
</init-param>
<init-param>
<param-name>username</param-name>
<param-value>admin</param-value> <= application login username
</init-param>
<init-param>
<param-name>password</param-name>
<param-value>sa</param-value> <= application login password
</init-param>
<init-param>
<param-name>sessionid</param-name>
<param-value>ELXSESSIONID</param-value> <= cookie name
(cookie name must be similar to what is
shown in RepertoireServer\config\jetty.xml)
</init-param>
<init-param>
<param-name>return</param-name>
<param-value>http://localhost:8079/RepertoireDemoServlet-7.2/ui/index.html</param-value>
<= This typically should be the main url link to the 3rd party application
</init-param>
....

- Deploy RepertoireSSO-7.x.x.war on your preferred application server (BEA Weblogic, Tomcat Web Server, JBoss Web Server etc) OR alternatively include the **ParamBasedRedirect** servlet class within your existing application.
- Once this is completed, the **RepertoireSSO** is now ready to be called upon within your application to access any resource in the Repertoire Server.

Session Cookie Handling during SSO process

The following describes the detail process of SSO login and logout ,in particular the management of the JSESSIONID cookie, which is required for proper login and logout. Developers implementing a SSO to Repertoire Server need to implement these steps in their programming language

- Logging in ....
    - User login to 3rd party application

    - The controlling server (e.g.**ParambasedRedirect**) should logon to ERS with user information using either **/logon.html** or BASIC AUTH and obtain a session cookie. (Default : JSESSIONID)
    - 3rd party application needs to hold this primary JSESSIONID within a user session. This can be done in Java using **javax.servlet.http.HttpSession**.
    - The controlling server then redirects the user to **/authenticate-session.html**, passing in the primary JSESSIONID as a parameter.
    - If the primary JSESSIONID is valid, the authentication details will be cloned into a new secondary JSESSIONID (note that this will not be the same value as the primary JSESSIONID) returned directly to the user (ie the browser).
    - Subsequent use of any resources with the ERS Server, would not require the intervention of the controlling server.

- Logging out...
    - The 3rd party application would need to set the primary JSESSIONID cookie back to the browser, in preparation of the logout process.
    - Issue a GET to **/logout.html**
    - Any user session that was authenticated based on the primary JSESSIONID will also be terminated.
    - For more information about accessing Repertoire Server resources, see documentation on REST API.

Integrating Repertoire Server in .Net

- Download DOTNET.zip
    - The above downloadable sample is meant to be used for Repertoire Server 7.1.x only.
    - It demonstrates how single-sign-on can be done using Rest API which is only available on Version 7.x.
    - For earlier versions - 5 and 6, view the following vb.net sample: .Net DLL which is using the ERSClient instead of Rest API.
- The following zipped file contains sample code which illustrates how to use single-sign-on in a .NET Framework.
    - RenderDasboard.aspx - How to render a Dashboard
    - RenderReport.aspx - How to render a Report using Rest API
    - RenderReport_tool.aspx - How to render a Report using Tool API

Note: Rendering a report using Tool API is meant for using the Server's extra functionality of unzipping the HTML and returning a reference to it. The REST API is for users who want to get the whole report back (eg. as a HTML zip file) and then uncompress it and return the reference to the user themselves. The Tool API will achieve this output and display the reports on the browser.

Rest API to invoke a URL -

- http://localhost:8080/report/ElixirSamples/Report/PetStore/ProductPerformance.rml

Tool API to invoke a URL -

- http://localhost:8080/tool/report?report=/ElixirSamples/Report/PetStore/ProductPerformance.rml

* The approach to single sign on in .Net is similar as compared to the Java sample above.

Tips and Approach Consideration

In RepertoireSSO-7.x.x.war, the userid and password was obtained from a configuration file. However, in real life deployment, the userid and password should be read from a more secure entity (database server or LDAP Server). Once a valid user is logged on to the main application, the same userid and password is used to authenticate at the Repertoire Server.

The RepertoireSSO-7.x.x.war sample provided here is a sample solution of how SSO can be achieved in Java web application. The same approach of achieving SSO can be done with any other programming languages.