

Integrating DaisyWiki and JAAS

WARNING

This is a draft version.
It is not meant to be complete and may contain mistakes, errors.
Your comments are welcome.

Introduction

Initially, Daisy Wiki has no support for JAAS based security. The lack of JAAS security support prohibits integrating of Daisy Wiki with other web applications. Luckily, adding JAAS security to the Daisy Wiki is not so difficult and can be easily performed. Daisy also offers a Repository subsystem, which can be utilized to store application users' credentials, whereas the Wiki has an administration console which can be used to easily manage user accounts.

Required steps

In order to get a working JAAS support in Daisy Wiki you might have to consider following steps:

1. Develop a custom JAAS login module and custom JAAS principal, in order to give other applications a possibility to get access to the information entered by the user on login page.
2. Change the way Wiki manages user credentials (actually the way it gets them).
3. In case, if you wish to use Repository as a user accounts store, you have to add corresponding authentication routines to your custom login module.

Further, these steps will be discussed in more detail.

Custom JAAS login module

A detailed explanation of what JAAS is, is on the official Sun [JAAS product page](#)¹. Quick start info on Apache Tomcat JAASRealm and custom module may be found in the "[Realm Configuration HOW-TO](#)"².

In our case, a new interface of the user principal was defined: SSOUserPasswordPrincipal:

```
public interface SSOUserPasswordPrincipal extends Principal {  
    public char[] getPassword();  
}
```

`java.security.Principal` already has a `getUser()` method defined. This interface will be shared among all applications using our custom SSO system, thus, we will put it into a separate JAR file, namely `sso-api.jar`. Our custom login module has to create user principals which will implement it, and web applications will use it in order to get user password.

-
1. <http://java.sun.com/products/jaas/>
 2. <http://tomcat.apache.org/tomcat-5.5-doc/realms-howto.html#JAASRealm>

Customizing Daisy Wiki

It seems that `org.outerj.daisy.frontend.WikiHelper` class is responsible for providing repository access to the rest of the frontend application. Thus, it will be used by us in order to insert JAAS specific code.

Due to a clever design (thanks, Bruno), in our case, only one method had to be changed: `getRepository(Request, ServiceManager)` for the `repository-is-null` branch:

```
if (repository == null) {
    Principal principal = request.getUserPrincipal();
    if (principal != null &&
        principal instanceof SSOUserPasswordPrincipal) {
        System.err.println("%% Principal is SSOUserPasswordPrincipal");
        SSOUserPasswordPrincipal tctPrincipal = (SSOUserPasswordPrincipal) principal;
        try {
            return login(tctPrincipal.getName(), new String(tctPrincipal.getPassword()),
                request, serviceManager);
        } catch (AuthenticationFailedException e) {
            e.printStackTrace();
            return getGuestRepository(serviceManager);
        }
    } else {
        System.err.println("%% Principal is not a SSOUserPasswordPrincipal: "
            + principal);
        return getGuestRepository(serviceManager);
    }
}
```

Of course, we have had also to add JAAS specific settings to `web.xml` descriptor:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected pages</web-resource-name>
    <url-pattern>/daisy/login</url-pattern>
    ...
  <auth-constraint>
    <role-name>JxTester</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description>no description</description>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>
/daisy/jaas_login.html
    </form-login-page>
    <!-- This should be uncommented to permit
login errors reporting. -->
    <form-error-page>
/daisy/logout
    </form-error-page>
  </form-login-config>
</login-config>

<security-role>
  <role-name>JxTester</role-name>
</security-role>
```

This way we have got a working JAAS support in Daisy Wiki. In order to minimize amount of work we have reused the login form. Setting logout action as a login error page makes it possible for the user to make a second try, as the HTTP session is invalidated and information about the first unsuccessful attempt is discarded. The site's skin will require some customizations, especially the login form (at least, user login and password input fields are to be removed, roles selector may remain).

Deployment issues on Tomcat

To avoid classloader issues, i would advice you to put `sso-api.jar` file into the `$CATALINA_HOME/common/endorsed` directory.

```
FIXME
TODO
```

Links

Author: Victor Anyakin

<mailto:anyakinvictor@yahoo.com>

Java Authentication and Authorization Service (JAAS)

<http://java.sun.com/products/jaas/>

Apache Tomcat Realm Configuration HOW-TO

<http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html#JAASRealm>

Fields

Name	Value
Category	Install & config