

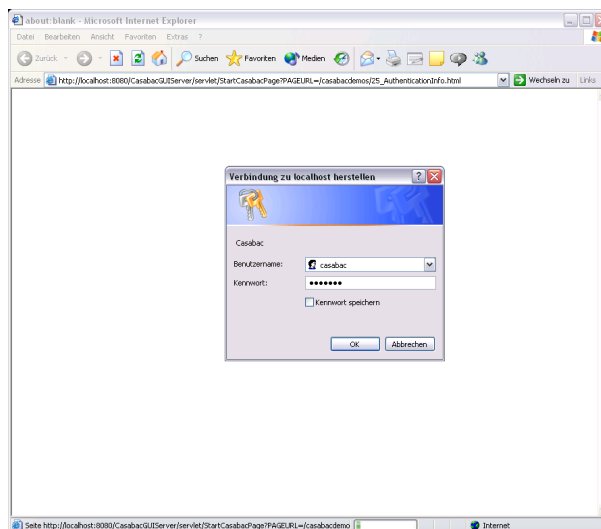
Using the Servlet Container's Authentication

This document shows an example how to link the servlet container's authentication services together with Casabac pages. The example is based on Tomcat 4.1.27 but should work with other versions as well.

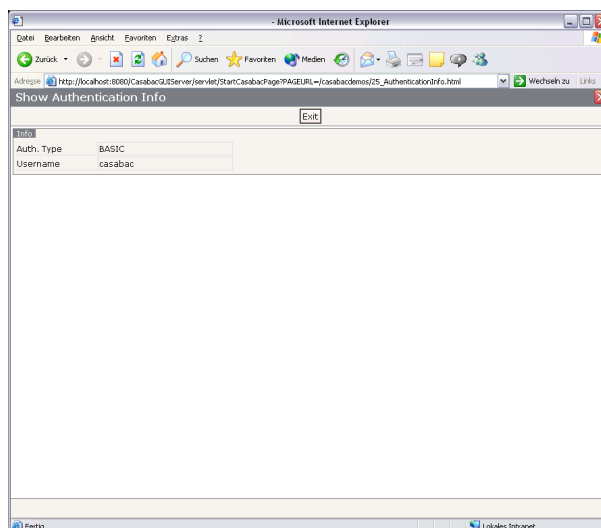
Goal

The following scenario will be demonstrates:

- When selecting a page from the CasabacGUIserver a popup shows up asking for authentication information. In the popup you specify a user and a password.



- After logging on the information will be used inside the Casabac adapter processing



Steps

Configure Tomcat

In the file "tomcat/conf/server.xml" you have to set up the way Tomcat checks authentication data. In other words: somewhere around there must be a list of users with their description (password, roles). This list needs to be accessed in order to check authentication and authorization of a login. There are various ways to do so - the easiest one will be chosen in this documentation: users will be checked against a definition in a certain file.

First set up the server.xml-configuration in the following way:

```

...
...
...
<!-- This Realm uses the UserDatabase configured in the global JNDI
resources under the key "UserDatabase". Any edits
that are performed against this UserDatabase are immediately
available for use by the Realm. -->
<!--
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
debug="0" resourceName="UserDatabase"/>
-->

<!-- Comment out the old realm but leave here for now in case we
need to go back quickly -->
<Realm className="org.apache.catalina.realm.MemoryRealm" />

<!-- Replace the above Realm with one of the following to get a Realm
stored in a database and accessed via JDBC -->

<!--
<Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
driverName="org.gjt.mm.mysql.Driver"
connectionURL="jdbc:mysql://localhost/authority"
connectionName="test" connectionPassword="test"
userTable="users" userNameCol="user_name" userCredCol="user_pass"
userRoleTable="user_roles" roleNameCol="role_name" />
-->
...
...
...

```

The "org.apache.catalina.realm.MemoryRealm" is chosen - the other ones are deactivated.

Then set up the file "tomcat/conf/tomcat-users.xml" in order to contain the roles required by the web application (description follows later on) and the user definitions:

```

<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="admin"/>
  <role rolename="manager"/>
  <role rolename="role1"/>
  <role rolename="casabacguiserver"/>
  <role rolename="tomcat"/>
  <user username="admin" password="" roles="admin,manager"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="casabac" password="casabac" roles="casabacguiserver"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
</tomcat-users>

```

Two lines were added: the new role "casabacguiserver" and the user "casabac". The user definition binds to the role definition.

Configure Web Application

The web application now needs to specify in which way authentication is to be checked. The following definition checks the access to all content of the web application.

```

...
...
...
<servlet-mapping>
  <servlet-name>StartMessagePage</servlet-name>
  <url-pattern>/servlet/StartMessagePage</url-pattern>
</servlet-mapping>

<security-constraint>
<web-resource-collection>
  <web-resource-name>Casabac</web-resource-name>
  <url-pattern>/*</url-pattern>
</web-resource-collection>
<auth-constraint>
  <role-name>casabacguiserver</role-name>
</auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Casabac</realm-name>
</login-config>
</web-app>

```

Please note that the role defined in "auth-constraint" is the one to map to the role defined in "tomcat-users.xml".

Access User Principal in Adapter

The User Principal of the logon is stored in accessible through the http-session-context:

```

package com.casabac.test25;

// This class is a generated one.

import java.security.Principal;
import java.util.*;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionContext;

import com.casabac.server.*;
import com.casabac.server.util.*;
import com.casabac.util.*;

public class AuthenticationInfoAdapter
  extends Model
{
  String m_authType;
  public String getAuthType() { return m_authType; }

  String m_userName;
  public String getUsername() { return m_userName; }

  /** initialisation - called when creating this instance*/
  public void init()
  {
    HttpServletRequest hsr = findHttpServletRequest();
    HttpSession hs = hsr.getSession();
    m_authType = hsr.getAuthType();
    Principal pr = hsr.getUserPrincipal();
    if (pr != null)
    {
      m_userName = pr.getName();
    }
  }
}

```

```
}
```

Variations

The example described above is the easiest and most straight forward way to make it work. Variations are:

- You can store user data in a database instead of in a file. In this case choose a different "<realm..."-definition in server.xml.
- You can choose between different types of authentication. "BASIC" was chosen in the example - a logon popup is coming up as consequence. You can e.g. write an own logon page (normal HTML page following certain conventions) and integrate it into form based authentication.
- You can define various different roles for you web application and build up a sophisticated authorization management.

Please have a look into the Servlet-specification for more information.