SSLWithFORMFallback7

This is another SSLWithFORMFallback class, this time for Tomcat 7. This is heavily reappropriated from the Tomcat 6 version by Vegar Neshaug

Gary

```
import java.io.IOException;
import java.security.Principal;
import java.security.cert.X509Certificate;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.http.HttpServletRequest;
import org.apache.catalina.Container;
import org.apache.catalina.Globals;
import org.apache.catalina.LifecycleException;
import org.apache.catalina.authenticator.AuthenticatorBase;
import org.apache.catalina.authenticator.Constants;
import org.apache.catalina.authenticator.FormAuthenticator;
import org.apache.catalina.authenticator.SSLAuthenticator;
import org.apache.catalina.connector.Request;
import org.apache.catalina.deploy.LoginConfig;
import org.apache.coyote.ActionCode;

/** Mostly borrowed from: http://wiki.apache.org/tomcat/SSLWithFORMFallback6
 *
 * @author Vegar Neshaug, minor edits for Tomcat7 by Gary Briggs
 */
public class SSLWithFORMFallback7 extends AuthenticatorBase {
    FormAuthenticator formAuthenticator = new FormAuthenticator();
    SSLAuthenticator sslAuthenticator = new SSLAuthenticator();

    @Override
    public boolean authenticate(Request rqst, javax.servlet.http.HttpServletResponse resp, LoginConfig lc)
throws IOException {
        // Have we already authenticated someone?)
        Principal principal = rqst.getUserPrincipal();
        //String ssoId = (String) request.getNote(Constants.REQ_SSOID_NOTE);
        if (principal != null) {
            // Associate the session with any existing SSO session in order
            // to get coordinated session invalidation at logout
            String ssoId = (String) rqst.getNote(Constants.REQ_SSOID_NOTE);
            if (ssoId != null) {
                associate(ssoId, rqst.getSessionInternal(true));
            }
            return (true);
        }

        // Get certificates from the request
        boolean certAuth = true;
        X509Certificate certs[] = (X509Certificate[]) rqst.getAttribute(Globals.CERTIFICATES_ATTR);
        if ((certs == null) || (certs.length < 1)) {
            rqst.getCoyoteRequest().action(ActionCode.REQ_SSL_CERTIFICATE, null);
            certs = (X509Certificate[]) rqst.getAttribute(Globals.CERTIFICATES_ATTR);
        }
        if ((certs == null) || (certs.length < 1)) {
            // No certificates
            certAuth = false;
        }

        // Delegate authentication request
        boolean retval;
        if (certAuth) {
            retval = sslAuthenticator.authenticate(rqst, resp, lc);
        } else {
            retval = formAuthenticator.authenticate(rqst, resp, lc);
        }
        // System.out.println("Retval: " + retval + ", certAuth: " + certAuth);
        return retval;
    }
```

```
    private String infoStr = null;
    @Override
    public String getInfo() {
        if(null == infoStr) {
            infoStr = this.getClass().getName();
        }
        return infoStr;
    }

    @Override
    protected String getAuthMethod() {
        return HttpServletRequest.CLIENT_CERT_AUTH;
//        return HttpServletRequest.FORM_AUTH;
    }

    @Override
    public void setContainer(Container container) {
        try {
            super.setContainer(container);
            sslAuthenticator.setContainer(container);
            formAuthenticator.setContainer(container);

            /* At time of writing, it appears .setContainer is the only
               thing necessary ahead of time to call .start() */
            formAuthenticator.start();
            sslAuthenticator.start();
        } catch (LifecycleException ex) {
            Logger.getLogger(SSLWithFORMFallback7.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```