SSLWithFORMFallback6

This is my implementation of SSL with FORM fallback. It has been tested on tomcat 6.0.16. Feel free to use at is you like. I do not know if this is the best approach, but it's simple and it works. Also see http://wiki.apache.org/tomcat/SSLWithFORMFallback

```
import java.io.IOException;
import java.security.Principal;
import java.security.cert.X509Certificate;
import javax.management.MalformedObjectNameException;
import javax.management.ObjectName;
import org.apache.catalina.Container;
import org.apache.catalina.Globals;
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.connector.Response;
import org.apache.catalina.deploy.LoginConfig;
import org.apache.coyote.ActionCode;
/**
 *
 * @author Vegar Neshaug
 */
public class SSLWithFormFallback extends AuthenticatorBase {
    // Our two authenticators
    FormAuthenticator formAuthenticator = new FormAuthenticator();
    SSLAuthenticator sslAuthenticator = new SSLAuthenticator();
    /**
     * Descriptive information about this implementation.
     */
    protected static final String info =
            "net.neshaug.SSLWithFormFallback/1.0";

    /**
     * Return descriptive information about this Valve implementation.
     */
    @Override
    public String getInfo() {
        return (info);
    }

    @Override
    public boolean authenticate(Request request, Response response, LoginConfig config) throws IOException {
        // Have we already authenticated someone?
        Principal principal = request.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) request.getNote(Constants.REQ_SSOID_NOTE);
            if (ssoId != null) {
                associate(ssoId, request.getSessionInternal(true));
            }
            return (true);
        }

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

        // Delegate authentication request
```

```
        if (certAuth) {
            return sslAuthenticator.authenticate(request, response, config);
        } else {
            return formAuthenticator.authenticate(request, response, config);
        }
    }

    @Override
    public void start() throws LifecycleException {
        super.start();
        formAuthenticator.start();
        sslAuthenticator.start();
    }

    @Override
    public void stop() throws LifecycleException {
        super.stop();
        formAuthenticator.stop();
        sslAuthenticator.stop();
    }

    // I'd rather not have the below, but it is necessary for the
    // authenticators to work properly.
    @Override
    public void setContainer(Container container) {
        super.setContainer(container);
        sslAuthenticator.setContainer(container);
        formAuthenticator.setContainer(container);
    }

    @Override
    public ObjectName createObjectName(String domain, ObjectName parent)
            throws MalformedObjectNameException {
        sslAuthenticator.createObjectName(domain, parent);
        formAuthenticator.createObjectName(domain, parent);
        return super.createObjectName(domain, parent);
    }
}
```

CategoryFAQ