[Jasny · web development]

Helping you out with PHP, MySQL & Javascript

Simple Single Sign-On for PHP (Ajax compatible)

by Arnold Daniels on 04/18/2009

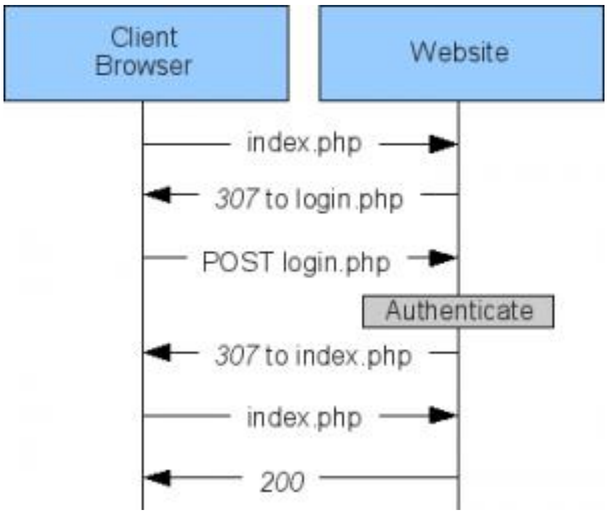
Associated websites often share user information, so a visitor only has to register once and can use that username and password for all sites. A good example for this is Google. You can use you google account for GMail, Blogger, iGoogle, google code, etc. This is nice, but it would be even nicer if logging in for GMail would mean I'm also logged in for the other websites. For that you need to implement single sign-on (SSO).

There are many single sign-on applications and protocols. Most of these are fairly complex. Applications often come with full user management solutions. This makes them difficult to integrate. Most solutions also don't work well with AJAX, because redirection is used to let the visitor log in at the SSO server.

I've written a simple single sign-on solution (400 lines of code), which works by linking sessions. This solutions works for normal websites as well as AJAX sites.

Without SSO

Let's start with a website that doesn't have SSO.



The client requests the index page. The page requires that the visitor is logged in. The server creates a new session and sends redirect to the login page. After the visitor has logged in, it displays the index page.

How it works

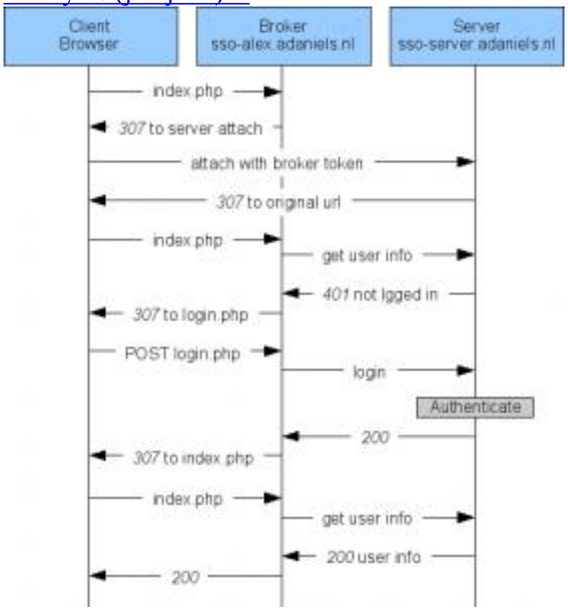
When using SSO, when can distinguish 3 parties:

- Client – This is the browser of the visitor
- Broker – The website which is visited
- Server – The place that holds the user information

The broker will talk to the server in name of the client. For that we want the broker to use the same session as the client. However the client won't pass the session id which it has at the server, since it's at another domain. Instead the broker will ask the client to pass a token to the server. The server uses the token, in combination with a secret word, to create a session key which is linked session of the client. The broker also know the token and the secret word and can therefore generate the same session key, which it uses to proxy login/logout commands and request info from the server.

First visit

-> Try it! (jan/jan1)<-



When you visit a broker website, it will check to see if a token cookie already exists. It it doesn't it, the broker sends a redirect to the server, giving the command to attach sessions and specifying the broker identity, a random token and the originally requested URL. It saves the token in a cookie.

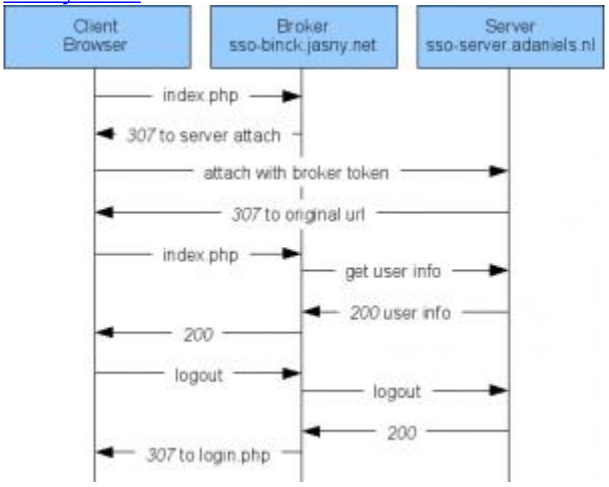
The server will generate a session key based on the broker identity, the secret word of the broker and the token and link this to the session of the client. The session key contains a checksum, so hackers can go out and use random session keys to grab session info. The server redirects the client back to the original URL. After this, the client can talk to the broker, the same way it does when not using SSO.

The client requests the index page at the broker. The page requires that the visitor is logged in. The broker generates the session key, using the token and the secret word, and request the user information at the server. The server responds to the broker that the visitor is not logged. The broker redirects the client to the login page.

The client logs in, sending the username and password to the broker. The broker sends the username and password to the server, again passing the session key. The server returns that login is successful to the broker. The broker redirects the client to the index page. For the index page, the broker will request the user information from the server.

Visiting another affiliate

-> Try it! <-



You visit another broker. It also checks for a token cookie. Since each broker is on their own domain, they have different cookies, so no token cookie will be found. The broker will redirect to the server attach to the user session.

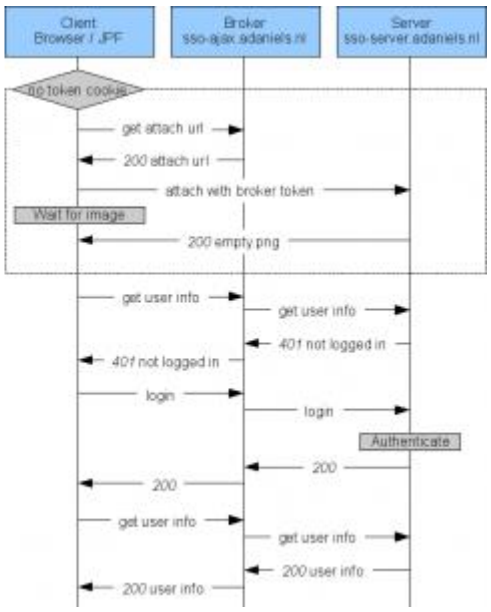
The server attaches a session key generated for this broker, which differs from the one for the first broker. It attaches it to the user session. This is the same session the first broker used. The server will redirect back to the original URL.

The client requests the index page at the broker. The broker will request user information from the server. Since the visitor is already logged in, the server returns this information. The index page is shown to the visitor.

Using AJAX / Javascript

-> Try it! <-
SSO and AJAX / RIA applications often don't go well together. With this type of application, you do not want to leave the page. The application is static and you get the data and do all actions through AJAX. Redirecting an AJAX call to a different website won't because of cross-site scripting protection within the browser.

With this solution the client only needs to attach the session by providing the server with a token generated by the broker. That attach request doesn't return any information. After attaching the client doesn't talk at all to the server any more. Authentication can be done as normal.



The client check for the token cookie. It it doesn't exists, he requests the attach URL from the broker. This attach url includes the broker name and the token, but not a original request URL. The client will open the received url in an <img> and wait until the image is loaded.

The server attaches the browser session key to the user session. When it's done it outputs a PNG image. When this image is received by the client, it knows the server has connected the sessions and the broker can be used for authentication. The broker will work as a proxy, passing commands and requests to the sso server and return results to the client.

To conclude

By connecting sessions, you give the broker the power to act as the client. This method can only be used if you trust all brokers involved. The login information is send through the broker, which he can easily store if the broker has bad intentions.

Don't be square, please share!