EPLconext Application

24/oct/2011; mdobrinic@cozmanova.com

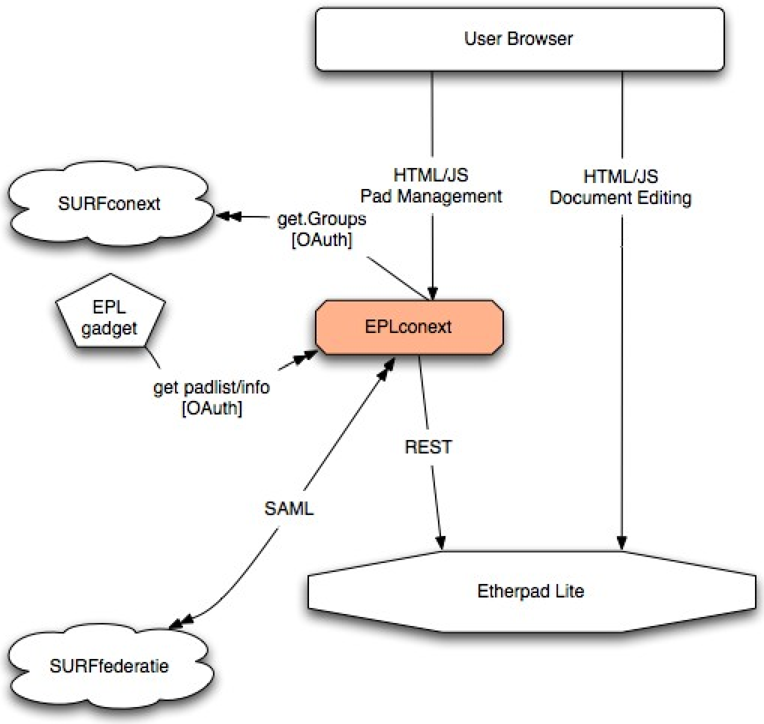
EPLconext is an application that wraps around EtherpadLite, adding team-context from SURFconext Portal and identity-context from SURFfederatie. All these functions are put together in a minimalistic application that provides a frontend for the user to interact with.

Part of the EPLconext application is integrated in an OpenSocial gadget, to enable a group-aware gadget that creates a bridge towards the main function of Etherpad Lite, collaborative editing of documents.

Functionality

Etherpad Lite is a collaborative document editor, and it provides this functionality without any front-end features for group and pad management whatsoever. It **does** offer an API to access these features, thereby enabling authenticated group- and user-support and (group)pad-management. The idea behind this API is that third parties can build their own authentication solutions and integration paths for Etherpad Lite to work in different environments. Like the SURFconext environment.

The authentication-, group- and pad-management functions are implemented in the EPLconext application. In an architectural overview picture, this looks like this:



- Authenticate using SURFconext (SURFfederatie)

The SAML SP functionality is based on SimpleSAMLphp.

- Retrieve group membership from SURFconext

Act as an OpenSocial-client and retrieve group information for the authenticated user, by using 3-legged OAuth.

- Provide gadget and REST service

Act as REST service, with services requiring authentication using either OAuth or a valid SimpleSAMLphp session.

- Manage Etherpad session context

Map authenticated users to Etherpad users and groups, and manage Etherpad Lite sessions in user's browser.

- Embed editor in group and pad selection view

Embed the Etherpad Lite editor in an iframe of the application, adding group- and pad-selection options to the user.

Implementation

The features that are listed in the previous paragraph, are all implemented in the EPLconext application. The main purpose of the EPLconext application, is to connect all the pieces together in different views for different purposes, both for human interaction as well as for service-consuming machine interaction. Different views are:

1) Application view for human clients
2) Service view, for providing data structures to machine clients
3) Gadget view, for embedding different views in OpenSocial context, for human clients

Application

The "human client"-application is made up of one view, implemented in the main.php script.

This script executes the following steps:

1. Establish user identity, using SimpleSAMLphp, for a given authsource definition
2. Establish user groups, using 3-legged OAuth of OpenSocial PHP-client (osapi-php)
   Note: made osapi-php include SimpleSAMLphp's OAuth library, as there are minor differences between the two, and version conflicts were present
3. Establish pads for the groups, using the EtherpadLiteClient PHP client to process the REST requests and responses
4. Render the user interface to show the groups, the pads per group, and a placeholder Iframe for the Etherpad editor.

All other functionality is executed in the webbrowser:

- Selecting the pad to edit
- Selecting the group, to show the grouppads and create a session-context for the selected group
- Create new pads in a group and Remove existing pads from a group

These functions are implemented using jQuery on top of an AJAX-architecture, using REST services to execute the server part of the requested functions.

Service

The Service application is implemented in padmanager.php. This script acts as a service broker. As such, it inspects how it was called, and dynamically instantiates the handler that can can service the request. The handlers are configured with contextual properties, like:

- is authentication required, and if so, does it require SimpleSAML or OAuth authentiation, or
- is this a group-related service, and if so, should group membership be enforced, or is it implicit
- etc. Etc

The response of the service application is always JSON.

Gadget

The gadget definition is an XML-script that specifies the behaviour of the OpenSocial gadget. It consists of 2 views:

1. home-view, rendered in one column of a column-layout
2. canvas-view, rendered in maximized area of the user interface

Home-view

The home-view is implemented in pure Gadget JavaScript. It requests a list of the pads of the current group. It can do this, by making an OAuth-authenticated call to the REST-service that returns the list of pads in the current group.

The identity of the user and the id of the group of the TAB that the gadget is part of, are both part of the (signed) OAuth request: the userid is implicitly derived from the access token, and the group-id is passed as 'opensocial_instance_id' parameter with the request. Therefore, the REST service can trust these values and return the requested information for the user in the group.

It does this in a JSON data structure, which is parsed and the result of which is rendered in the gadget home-view.

Canvas-view

The original thought would be to make the home-view the user interface where a user would click on a pad, which would trigger the gadget to switch to the canvas-view where the Etherpad Lite editor would show the selected pad for editting.

Too bad this is not possible, because of the fact that view-switching as described, is not supported by the SURFconext OpenSocial-container.

Instead, the canvas-view shows the list of pads in the same way that the home-view does, but selecting a pad results in the editor to be loaded in the gadget-iframe.

The canvas-view is no JavaScript-gadget. Instead, it is an IFrame-gadget, that is included **with OAuth signature**. As such, the user identity and group id are provided to the script that is executed in the IFrame, and the pad-list can be rendered immediately without having to double-check the user's identity or whether the user is actually member of the provided group.

Note that the content of the request (the request_body) contains the result of the <os:ViewerRequest .... />-element. This data is also authentic, as there is a oauth_body_hash-parameter calculated over the body that is verified, and this body_hash-parameter is also part of the signed data.

The result of the pad-listing-script is rendered through OpenSocial, that acts as a proxy. This means that is is hosted from the surfconext.nl-domain. When selecting the pad though, the script is executed from its own domain, and no longer in the OpenSocial-container domain. This means that is is **not** possible to use a generic session mechanism to store the user identity and group information that were passed to the script initially. For this purpose, a one-time token is added to the link that opens the Etherpad Lite editor, to be able to transfer the identity information.