Installation des portfolios sur Plone (3.1 RC)

Thomas van Oudenhove — PraKsys

30 avril 2008

Table des matières

1 Installation

Ce tutoriel est testé pour la version 3.1 de Plone, alors en *Release Candidate*.
Télécharger la version sur `http://www.plone.org`.
Installer Plone (les instructions détaillées sont dans le fichier `README.txt` une fois l'archive extraite.

2 Les portfolios

2.1 Installation

Télécharger et installer le produit ATBookmark (disponible sur le site de Plone, dans la catégorie *« Add-on Products »*.

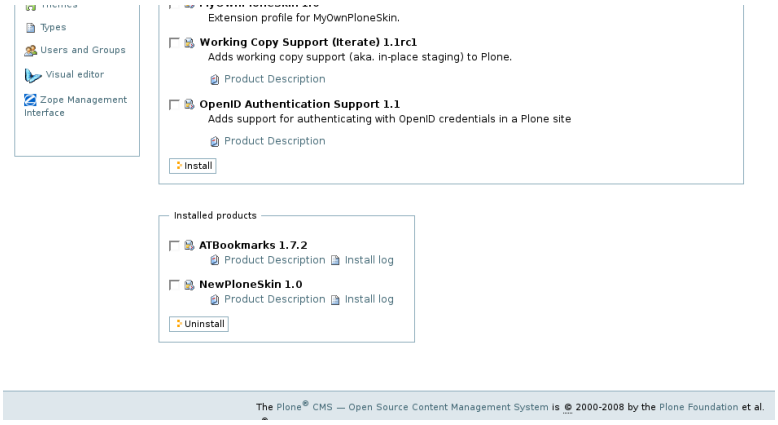

FIG. 1 – Le produit « ATBookmark » est installé

Dans Plone, créer un *Large Plone Folder* (par exemple, « portfolio ») qui est destiné à contenir tous les portfolios.

2.2 Règle de gestion

Dans la *Plone Management Interface*, assigner une règle de gestion (*Content Rules*).
Activer cette règle globalement et à la racine du site.

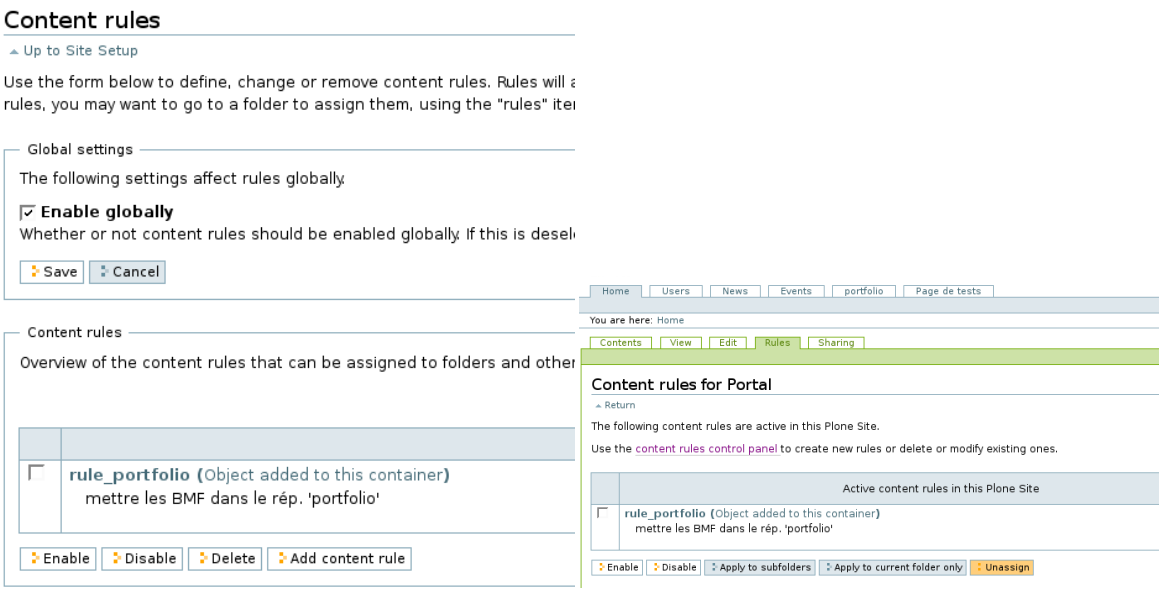

FIG. 2 – Double activation de la règle de gestion

2.3 *Workflow*

Dans la *Zope Management Interface*, dans portal_workflow, copier le *workflow simple_publication_workflow*, le renommer en *portfolio_workflow* et affecter ce nouveau *workflow* au type « BookmarkFolder ».

Modifier ensuite ce *workflow* : ne laisser que les états *visible* et *published* (avec *visible* comme état initial). Pour les transitions, ne laisser ensuite que les transitions *publish* et *retract*.

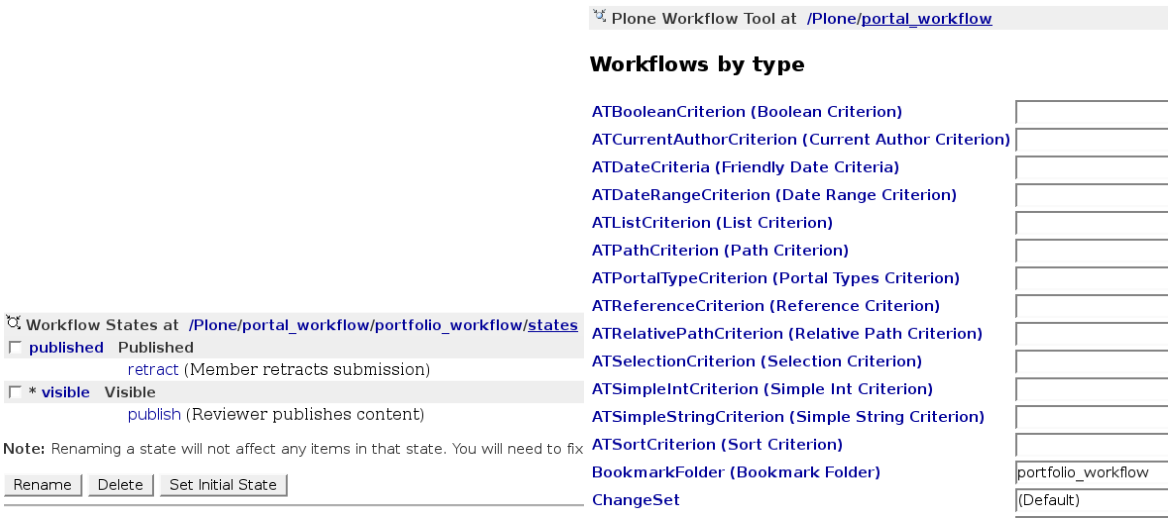

FIG. 3 – Nouveau *workflow* et affectation à *Bookmark Folder*

3 Les portlets

Dans la *Zope Management Interface*, dans portal_skins, ajouter deux *Page Template* pour créer les portlets d'affichage des portfolios.

3.1 Les portfolios publiés

Le premier *Page Template*, pour afficher les portfolios publiés contient le code suivant :

```
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal"
      i18n:domain="plone">
<body>

<div metal:define-macro="portlet"
     tal:define="results python:request.get('portfolios',
                                           here.portal_catalog.searchResults(portal_type='BookmarkFolder',
                                                                              sort_on='Date',
                                                                              sort_order='reverse',
                                                                              review_state='published')[:3]);
                ">

    <div class="portlet"
         id="portlet-portfolios">

        <div class="portletHeader">
            <span>Derniers portfolios publiés</span>
        </div>

        <div class="portletBody">

            <tal:block tal:repeat="obj results">

                <div tal:define="oddrow repeat/obj/odd"
                     tal:attributes="class python:test(oddrow, 'portletContent even', 'portletContent odd')">

                    <a href=""
                       tal:attributes="href obj/getURL;
                                       title obj/Description">
                        <tal:block replace="structure here/newsitem_icon.gif"/>
                        <span tal:replace="python:test(obj.Title, obj.Title, obj.getId)"> Extended Calendar
                            Product
                        </span>
                    </a>

                    <div class="portletDetails" align="right"
                         tal:content="python:here.toLocalizedTime(obj.Date)">July 7, 08:11
                    </div>
                </div>

            </tal:block>

        </div>

        <div class="portletFooter">
            <a href="/Plone/portfolio/">See all of'em</a>
        </div>

    </div>

</div>

</body>
</html>
```

Le nommer par exemple *portfolio_published* et sauvegarder.

3.2 Mes portfolios

Le deuxième *Page Template*, pour afficher les portfolios appartenant à l'utilisateur connecté, contient le code suivant :

```
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal"
      i18n:domain="plone">
<body>

<div metal:define-macro="portlet"
```

```
tal:define="userName␣user/getUserName;
␣␣␣␣␣␣␣␣␣␣␣results␣python:request.get('portfolios',␣here.portal_catalog.searchResults(␣portal_type='
    BookmarkFolder'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣,␣sort_on='Date'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣,␣sort_order='reverse'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣,␣Creator=userName)[:5]);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣">

    <div class="portlet"
         id="portlet-portfolios">

        <div class="portletHeader">
            <span>Mes portfolios</span>
        </div>

        <div class="portletBody">

            <tal:block tal:repeat="obj␣results">

                <div tal:define="oddrow␣repeat/obj/odd"
                     tal:attributes="class␣python:test(oddrow,␣'portletContent␣even',␣'portletContent␣odd')">

                    <a href=""
                       tal:attributes="href␣obj/getURL;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣title␣obj/Description">
                        <tal:block replace="structure␣here/newsitem_icon.gif"/>
                        <span tal:replace="python:test(obj.Title,␣obj.Title,␣obj.getId)"> Extended Calendar
                             Product </span>
                    </a>

                    <div class="portletDetails" align="right"
                         tal:content="python:obj.review_state">State
                    </div>
                </div>

            </tal:block>

        </div>

    </div>

</div>

</body>
</html>
```

Le nommer par exemple *portfolio_by_author* et sauvegarder.

3.3 Gestion des portlets

Sur le site Plone, cliquer sur le lien *Manage Portlets* et ajouter un *Classic portlet*. Dans le champ *Template*, mettre l'identifiant du *Page Template* développé ci-dessus. Refaire l'opération pour le deuxième portlet.

4 L'ajout d'un lien par le portlet de recherche

Le principe est le suivant : on ajoute un portlet de recherche avec la *livesearch*. D'une page dont on veut garder le lien dans un portfolio, on commence à remplir le champ ; on clique alors sur le portfolio auquel on veut ajouter un lien vers cette page. Le lien est alors automagiquement ajouté au portfolio en question.

Pour ajouter le portlet de recherche, aller dans *Manage Portlets*. Une fois le portlet ajouté, il faut l'adapter. Pour ce faire, aller dans la *Zope Management Interface*, dans portal_view_customizations. Chercher le portlet *search* (search.pt) et cliquer sur *customize*. Changer le code pour qu'il contienne le code ci-dessous et sauvegarder.

```
<dl class="portlet␣portletSearch"
    i18n:domain="plone">

  <dt class="portletHeader">
      <span class="portletTopLeft"></span>
      <a class="tile"
         tal:attributes="href␣view/search_form"
         i18n:translate="box_search">Add to portfolio</a>
```

```
      <span class="portletTopRight"></span>
  </dt>

    <dd class="portletItem">
      <form name="searchform" action="search"
            tal:attributes="action view/search_action">
        <div class="LSBox">
          <input class="searchField inputLabel"
              name="SearchableText"
              type="text"
              size="15"
              title="Search Portfolio"
              i18n:attributes="title title_search_title;"
              tal:attributes="value request/SearchableText|nothing;
                              class python:'inputLabel searchField portlet-search-gadget'"
              />

          <div class="LSResult" style="">
            <div class="LSShadow"></div>
          </div>
        </div>
      </form>
      <div class="visualClear"><!-- --></div>
    </dd>
</dl>
```

Dans le *filesystem*, rechercher les fichiers suivants :

`search.py` : ici, dans `/opt/Plone-3.1/buildout-cache/eggs/plone.app.controlpanel-1.0.5-py2.4.egg/plone/app/controlpanel/search.py`

`createObject.cpy` : ici, dans `/opt/Plone-3.1/zinstance/parts/plone/CMFPlone/skins/plone_scripts/`

`livesearch_reply.py` : ici, dans `/opt/Plone-3.1/zinstance/parts/plone/CMFPlone/skins/plone_scripts/`

Modifier `livesearch_reply.py` pour que le code devienne :

```
## Script (Python) "livescript_reply"
##bind container=container
##bind context=context
##bind namespace=
##bind script=script
##bind subpath=traverse_subpath
##parameters=q, limit=10, path=None, docurl=None, doctitle=None
##title=Determine whether to show an id in an edit form

from Products.CMFCore.utils import getToolByName
from Products.CMFPlone import PloneMessageFactory as _
from Products.CMFPlone.browser.navtree import getNavigationRoot
from Products.CMFPlone.utils import safe_unicode
from Products.PythonScripts.standard import url_quote
from Products.PythonScripts.standard import url_quote_plus
from Products.PythonScripts.standard import html_quote

ploneUtils = getToolByName(context, 'plone_utils')
portal_url = getToolByName(context, 'portal_url')()
pretty_title_or_id = ploneUtils.pretty_title_or_id
plone_view = context.restrictedTraverse('@@plone')

portalProperties = getToolByName(context, 'portal_properties')
siteProperties = getattr(portalProperties, 'site_properties', None)
useViewAction = []
if siteProperties is not None:
    useViewAction = siteProperties.getProperty('typesUseViewActionInListings', [])

# SIMPLE CONFIGURATION
USE_ICON = True
USE_RANKING = False
MAX_TITLE = 29
MAX_DESCRIPTION = 93

# generate a result set for the query
catalog = context.portal_catalog

friendly_types = ploneUtils.getUserFriendlyTypes()
```

```
def quotestring(s):
    return '"%s"' % s

def quote_bad_chars(s):
    bad_chars = ["(", ")"]
    for char in bad_chars:
        s = s.replace(char, quotestring(char))
    return s

# for now we just do a full search to prove a point, this is not the
# way to do this in the future, we'd use a in-memory probability based
# result set.
# convert queries to zctextindex

# XXX really if it contains + * ? or -
# it will not be right since the catalog ignores all non-word
# characters equally like
# so we don't even attept to make that right.
# But we strip these and these so that the catalog does
# not interpret them as metachars
##q = re.compile(r'[\*\?\-\+]+').sub(' ', q)
for char in '?-+*':
    q = q.replace(char, ' ')
r=q.split()
r = " AND ".join(r)
r = quote_bad_chars(r)+'*'
searchterms = url_quote_plus(r)

site_encoding = context.plone_utils.getSiteEncoding()
if path is None:
    path = getNavigationRoot(context)
# TvO
#results = catalog(SearchableText=r, portal_type=friendly_types, path=path)
results = catalog(SearchableText=r, portal_type='BookmarkFolder', path=path)

searchterm_query = '?searchterm=%s'%url_quote_plus(q)

RESPONSE = context.REQUEST.RESPONSE
RESPONSE.setHeader('Content-Type', 'text/xml;charset=%s' % site_encoding)

# replace named entities with their numbered counterparts, in the xml the named ones are not correct
#   &darr;      --> &#8595;
#   &hellip;    --> &#8230;
legend_livesearch = _('legend_livesearch', default='LiveSearch &#8595;')
label_no_results_found = _('label_no_results_found', default='No matching results found.')
label_advanced_search = _('label_advanced_search', default='Advanced Search&#8230;')
label_show_all = _('label_show_all', default='Show all&#8230;')

ts = getToolByName(context, 'translation_service')

output = []

def write(s):
    output.append(safe_unicode(s))


if not results:
    write('''<fieldset class="livesearchContainer">''')
    write('''<legend id="livesearchLegend">%s</legend>''' % ts.translate(legend_livesearch))
    write('''<div class="LSIEFix">''')
    write('''<div id="LSNothingFound">%s</div>''' % ts.translate(label_no_results_found))
    write('''<div class="LSRow">''')
    write('<a href="search_form" style="font-weight:normal">%s</a>' % ts.translate(label_advanced_search))
    write('''</div>''')
    write('''</div>''')
    write('''</fieldset>''')

else:
    write('''<fieldset class="livesearchContainer">''')
    write('''<legend id="livesearchLegend">%s</legend>''' % ts.translate(legend_livesearch))
    write('''<div class="LSIEFix">''')
    write('''<ul class="LSTable">''')
    for result in results[:limit]:
```

```
        icon = plone_view.getIcon(result)
        itemUrl = result.getURL()
        if result.portal_type in useViewAction:
            itemUrl += '/view'
#         itemUrl = itemUrl + searchterm_query
#         bmf_id = getToolByName(result, 'plone_utils').getId()
        link_title = doctitle.decode('utf-8')
#         regexp = re.compile(' &mdash; Portal')
#         link_title = regexp.sub( '', my_title)
        link_desc = "TODO"
        link_url = url_quote(docurl)

        write('''<li class="LSRow">''')
        if icon.url is not None and icon.description is not None:
            write('''<img src="%s" alt="%s" width="%i" height="%i" />''' % (icon.url,
                                                                            icon.description,
                                                                            icon.width,
                                                                            icon.height))
        full_title = safe_unicode(pretty_title_or_id(result))
        if len(full_title) > MAX_TITLE:
            display_title = ''.join((full_title[:MAX_TITLE],'...'))
        else:
            display_title = full_title
        full_title = full_title.replace('"', '&quot;')
#         write('''<a href="%s" title="%s">%s</a>''' % (itemUrl, full_title, display_title))
        write('''<a href="%s/createLink?title=%s&description=%s&remote_url=%s&bmf=%s">%s</a>''' % (itemUrl,
             link_title, link_desc, link_url, full_title, display_title))
#         write('''<a href="%s/portfolio/%s/createLink" title="%s">%s</a>''' % (itemUrl, full_title, full_title,
    display_title))
        write('''<span class="discreet">[%s%%]</span>''' % result.data_record_normalized_score_)
        display_description = safe_unicode(result.Description)
        if len(display_description) > MAX_DESCRIPTION:
            display_description = ''.join((display_description[:MAX_DESCRIPTION],'...'))
        # need to quote it, to avoid injection of html containing javascript and other evil stuff
        display_description = html_quote(display_description)
        write('''<div class="discreet" style="margin-left: 2.5em;">%s</div>''' % (display_description))
        write('''</li>''')
        full_title, display_title, display_description = None, None, None

    write('''<li class="LSRow">''')
    write( '<a href="search_form" style="font-weight:normal">%s</a>' % ts.translate(label_advanced_search))
    write('''</li>''')

    if len(results)>limit:
        # add a more... row
        write('''<li class="LSRow">''')
        write( '<a href="%s" style="font-weight:normal">%s</a>' % ('search?SearchableText=' + searchterms, ts.
             translate(label_show_all)))
        write('''</li>''')
    write('''</ul>''')
    write('''</div>''')
    write('''</fieldset>''')

return '\n'.join(output).encode(site_encoding)
```

Copier `createObject.cpy` en `createLink.cpy`, et changer le code pour qu'il devienne :

```
## Controller Python Script "createLink"
##bind container=container
##bind context=context
##bind namespace=
##bind script=script
##bind state=state
##bind subpath=traverse_subpath
##parameters=id=None, type_name='Link', script_id=None, title=None, description=None, remote_url=None, bmf=None
##title=
##

from DateTime import DateTime
from Products.CMFPlone.utils import transaction_note
from Products.CMFPlone import PloneMessageFactory as _
from Products.CMFCore.utils import getToolByName
```

```
RESPONSE = context.REQUEST.RESPONSE

# adapter la chaîne de remplacement à votre cas...
link_url = remote_url.replace('http://localhost:8080', '')
link_title = title.replace(' - Portal', '')

new_id = 'bmf-' + link_url.replace('/', '-')

portfolio = getattr(context.portfolio, bmf, None)

createdlink = portfolio.invokeFactory(id=new_id, type_name='Link')
portfolio[new_id].edit(title=link_title,
                       remote_url=link_url,
                       description=description)

return RESPONSE.redirect('/Plone/portfolio/' + bmf + '/' + new_id + '/edit')
```

Enfin, dans le fichier `search.py`, commenter les lignes 63 et 66 (qui doivent contenir une instruction du type `self.jstool.getResource('livesearch.js').setEnabled(XXXX)`) et ajouter une ligne 67 pour activer la *livesearch*. La fonction `set_enable_livesearch` devient donc :

```
60      def set_enable_livesearch(self, value):
61          if value:
62              self.context.manage_changeProperties(enable_livesearch=True)
63  #            self.jstool.getResource('livesearch.js').setEnabled(True)
64          else:
65              self.context.manage_changeProperties(enable_livesearch=False)
66  #            self.jstool.getResource('livesearch.js').setEnabled(False)
67          self.jstool.getResource('livesearch.js').setEnabled(True)
68          self.jstool.cookResources()
```

Une fois tous ces fichiers changés, redémarrer Zope.

Enfin, dans la *Zope Management Interface*, dans portal_skins/plone_ecmascript, adapter `livesearch.js` (*customize*) pour que le code devienne :

```
var livesearch = function (){

    // Delay in milliseconds until the search starts after the last key was
    // pressed. This keeps the number of requests to the server low.
    var _search_delay = 400;
    // Delay in milliseconds until the results window closes after the
    // searchbox looses focus.
    var _hide_delay = 400;

    // stores information for each searchbox on the page
    var _search_handlers = {};

    // constants for better compression
    var _LSHighlight = "LSHighlight";

    function _searchfactory($form, $inputnode) {
        // returns the search functions in a dictionary.
        // we need a factory to get a local scope for the event, this is
        // necessary, because IE doesn't have a way to get the target of
        // an event in a way we need it.
        var $lastsearch = null;
        var $request = null;
        var $cache = {};
        var $querytarget = "livesearch_reply";
        if (typeof portal_url != "undefined")
            $querytarget = portal_url + "/" + $querytarget;
        var $$result = $form.find('div.LSResult');
        var $shadow = $form.find('div.LSShadow');
        var $path = $form.find('input[name=path]');

        function _hide() {
            // hides the result window
            $$result.hide();
            $lastsearch = null;
        };

        function _hide_delayed() {
            // hides the result window after a short delay
            window.setTimeout(
```

```
            'livesearch.hide("' + $form.attr('id') + '")',
            _hide_delay);
    };

    function _show($data) {
        // shows the result
        $$result.show();
        $shadow.html($data);
    };

    function _search() {
        // does the actual search
        if ($lastsearch == $inputnode.value) {
            // do nothing if the input didn't change
            return;
        }
        $lastsearch = $inputnode.value;

        if ($request && $request.readyState < 4)
            // abort any pending request
            $request.abort();

        // Do nothing as long as we have less then two characters -
        // the search results makes no sense, and it's harder on the server.
        if ($inputnode.value.length < 2) {
            _hide();
            return;
        }

        var $$query = { q: $inputnode.value };
        if ($path.length && $path[0].checked)
            $$query['path'] = $path.val();
        // turn into a string for use as a cache key
        $$query = jq.param($$query);

        // check cache
        if ($cache[$$query]) {
            _show($cache[$$query]);
            return;
        }

        // the search request (retrieve as text, not a document)
        $request = jq.get($querytarget, $$query, function($data) {
            // show results if there are any and cache them
            _show($data);
            $cache[$$query] = $data;
        }, 'text');
    };

    function _search_delayed() {
        // search after a small delay, used by onfocus
        window.setTimeout(
            'livesearch.search("' + $form.attr('id') + '")',
            _search_delay);
    };

    return {
        hide: _hide,
        hide_delayed: _hide_delayed,
        search: _search,
        search_delayed: _search_delayed
    };
};

function _keyhandlerfactory($form) {
    // returns the key event handler functions in a dictionary.
    // we need a factory to get a local scope for the event, this is
    // necessary, because IE doesn't have a way to get the target of
    // an event in a way we need it.
    var $timeout = null;
    var $$result = $form.find('div.LSResult');
    var $shadow = $form.find('div.LSShadow');
```

```
    function _keyUp() {
        // select the previous element
        $cur = $shadow.find('li.LSHighlight').removeClass(_LSHighlight);
        $prev = $cur.prev('li');
        if (!$prev.length) $prev = $shadow.find('li:last');
        $prev.addClass(_LSHighlight);
        return false;
    };

    function _keyDown() {
        // select the next element
        $cur = $shadow.find('li.LSHighlight').removeClass(_LSHighlight);
        $next = $cur.next('li');
        if (!$next.length) $next = $shadow.find('li:first');
        $next.addClass(_LSHighlight);
        return false;
    };

    function _keyEscape() {
        // hide results window
        $shadow.find('li.LSHighlight').removeClass(_LSHighlight);
        $$result.hide();
    };

    function _handler($event) {
        // dispatch to specific functions and handle the search timer
        window.clearTimeout($timeout);
        switch ($event.keyCode) {
            case 38: return _keyUp();
            case 40: return _keyDown();
            case 27: return _keyEscape();
            case 37: break; // keyLeft
            case 39: break; // keyRight
            default: {
                $timeout = window.setTimeout(
                    'livesearch.search("'␣+␣$form.attr('id')␣+␣'")',
                    _search_delay);
            }
        }
    };

    function _submit() {
        // check whether a search result was selected with the keyboard
        // and open it
        var $target = $shadow.find('li.LSHighlight a').attr('href');
        if (!$target) return;
        window.location = $target;
        return false;
    };

    return {
        handler: _handler,
        submit: _submit
    }
};

function _setup(i) {
    // add an id which is used by other functions to find the correct node
    var $id = 'livesearch' + i;
    var $form = jq(this).parents('form:first');
    var $key_handler = _keyhandlerfactory($form);
    _search_handlers[$id] = _searchfactory($form, this);

    $form.attr('id', $id).css('white-space', 'nowrap').submit($key_handler.submit);
    jq(this).attr('autocomplete','off')
            .keydown($key_handler.handler)
            .focus(_search_handlers[$id].search_delayed)
            .blur(_search_handlers[$id].hide_delayed);
};

jq(function() {
    // find all search fields and set them up
    jq("#searchGadget,input.portlet-search-gadget").each(_setup);
```

```
    });

    return {
        search: function(id) {
            _search_handlers[id].search();
        },
        hide: function(id) {
            _search_handlers[id].hide();
        }
    };
}();
```

Le résultat est le suivant :

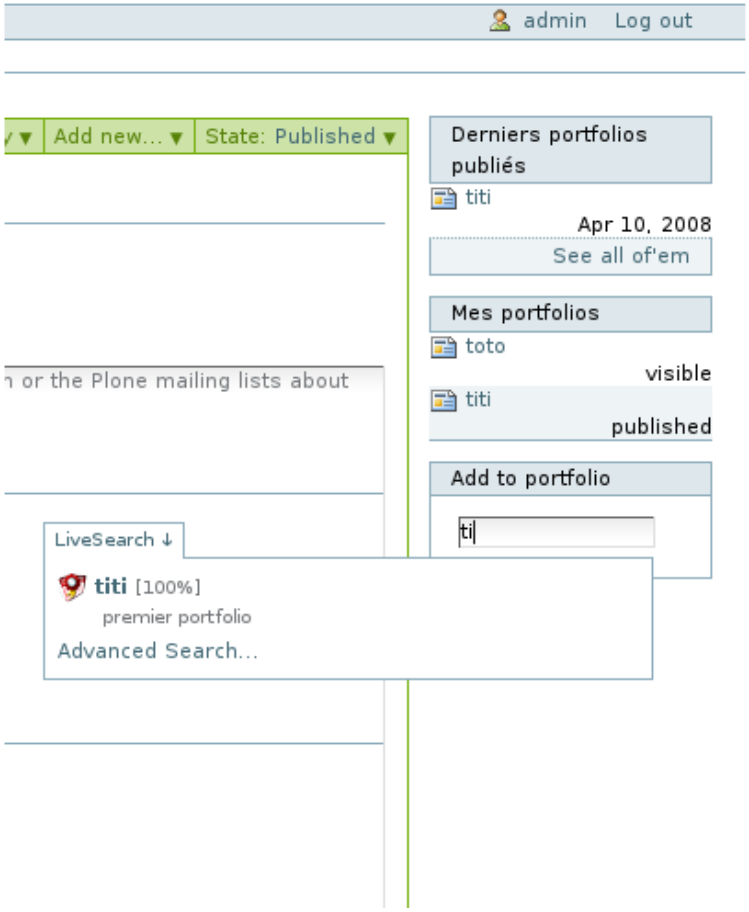

FIG. 4 – Aperçu des 3 portlets créés pour gérer les portfolios

Cliquer sur le portfolio retourné ajoute alors un lien vers la page courante dans ce portfolio. L'utilisateur n'a plus qu'à rentrer la description et éventuellement modifier le titre.