Adding Portlets To eduCommons

A portlet is a user-interface component that can be "plugged-in" to the architecture of Plone, the framework upon which eduCommons is based. There are several portlets that are natively available when you install eduCommons: the Course Info portlet that appears in the right navigation is an example, the Course Builder portlet is another. This tutorial will explain how portlets can be added and removed and how you can create your own portlet from scratch.

Adding and Removing Existing Portlets

If a portlet already exists that you either want to remove or add, follow these steps.

- navigate to the site as an administrator.
- click on the *manage portlets* link.

The *manage portlets* page allows you to add and remove portlets.

- To add a portlet click on the *Add portlet...* drop down menu and select the portlet to add.
- To delete a portlet click on the small red x next to the portlet to remove.

You can also change the order that the portlets appear by using the small blue arrows next to the porlets.

Creating Your Own Portlet

One of the simplest ways to create a portlet is to use the ZopeSkel tool to create a skeleton product that you can then modify for your own purposes. The Plone site has an excellent article describing how to install and use the Zope Once you have ZopeSkel installed you can create a new portlet package using the plone3_portlet template.

```
paster create -t plone3_portlet
```

Enter the information, as prompted, for the portlet you are creating. To select the defaults, just press enter.

```
Enter project name: SamplePortletProject
Variables:
  egg:      SamplePortletProject
  package:  sampleportletproject
  project:  SamplePortletProject
Enter namespace_package (Namespace package (like plone)) ['collec-
tive']: sample_portlet_project
Enter namespace_package2 (Nested namespace package (like app)) ['portlet']:
Enter package (The package contained namespace package (like exam-
ple)) ['example']:
Enter zope2product (Are you creating a Zope 2 Product?) [True]:
Enter version (Version) ['0.1']:
Enter description (One-
line description of the package) ['']: This is a sample portlet product
Enter long_description (Multi-line description (in reST)) ['']:
Enter author (Author name) ['Plone Foundation']:
Enter author_email (Author email) ['plone-
developers@lists.sourceforge.net']:
Enter keywords (Space-separated keywords/tags) ['']:
Enter url (URL of homepage) ['http://plone.org']:
Enter license_name (License name) ['GPL']:
Enter zip_safe (True/False: if the package can be dis-
tributed as a .zip file) [False]:
Enter portlet_name (Portlet name (human readable)) ['Example port-
let']: Sample Portlet
Enter portlet_type_name (Portlet type name (should not contain spaces)) ['Ex-
amplePortlet']: SamplePortlet
```

A portlet package will be created in the folder where you ran the script. Follow the directions in the Plone article to install the new portlet package using easy_install. Another option is to tell the paster script to create a Plone 2 product and then copy this product into the Products directory of your plone instance (the Plone 2 product is buried down a few levels in the package). In either case, after you restart your Plone instance and navigate to the *Add-on Products* page of your Plone site you will see the new product listed. After installing the product into your Plone site you will be able to manage it as you do the other portlets in your site.

Customizing Portlets

If you look at the code, generated in the last section, you will see that this new product is relatively simple. It contains a configure.zcml file, a template file, a python file, a profile folder, and a tests folder. Often the only places that will need to be changed to customize the portlet will be the template file and the renderer within the python file. Here are some examples of portlets that do just that:

OerRecommender Portlet

This is the oerrecommender.pt template file for the oerrecommender. The file consists of a script tag that retrieves a series of recommendations from oerrecommender.org.

```
<div id="recommendations">
  <script language="JavaScript"
          tal:define="url here/absolute_url"
```

```
          tal:attributes="src string:http://www.oerrecommender.org/recommendations.pjs?educommons=
          type="text/javascript">
  </script>
</div>
```

Here is the renderer for the oerrecommenderportlet.py file.

```
class Renderer(base.Renderer):
    """ Render the OER Portlet """
    render = ViewPageTemplateFile('oerrecommender.pt')

    def __init__(self, context, request, view, manager, data):
        super(Renderer, self).__init__(context, request, view, man-
ager, data)
        self.props = self.context.portal_properties.educommons_properties

    @property
    def available(self):
        return self.props.oerrecommender_enabled
```

You'll notice that the renderer is where we define which page template to use when rendering the template. This file also contains a property that allows the portlet to be conditionally displayed based on whether the oerrecommender is enabled.

CourseBuilder Portlet

The Course Builder tool allows users to quickly create a course with its associated files and departments. In this coursebuilder.pt template file you can see the portlet displaying a link to the Course Builder form (the '@@' symbol signifies that the resource is a view).

```
<dl class="portlet portletCourseBuilder"
    i18n:domain="eduCommons">

  <dt class="portletHeader"
      i18n:translate="box_course_builder">
    Course Builder
  </dt>

  <dd class="portletItem even">
    <a href=""
         i18n:translate="text_build_course"
         tal:attributes="href string:${here/portal_url}/@@coursebuilderform">Build a Course</a>
  </dd>

</dl>
```

CourseInfo Portlet

The CourseInfo portlet displays course statistics for the objects within each course. Methods are defined with the python file that are then used with the template file. For example, the renderer for the portlet in the courseinfoportlet.py file contains several methods

```
...
def statePercent(self, state=''):
  """ return the percent value for a given state """

  if 0 == self.total:
    return '0%'
  else:
    width = float(self.stateCounts[state])/float(self.total) * 100.0
    return '%d%%' %int(width)
...
```

that are then used in the courseinfo.pt template file.

```
...
    <img alt="barchart" class="statecolumnimage"
                tal:attributes="width python:view.statePercent(item);
                src string:bargraphic.gif" />
...
```

Additional Resources

There are some great resources to help you in customizing existing portlets and for adding functionality to portlets that you've created using the ZopeSkel product.

If you would like to do some quick and dirty customization through-the-web you'll probably want to use the portal_view_customizations tool in the ZMI. It's similar to portal_skins but for view, viewlets, and portlets.

Martin Aspeli has a whole tutorial on customizing a plone site which includes a section on customizing portlet renderers.

The Plone site also has a tutorial that shows how to override existing portlets.

View document source. Generated on: 2009-01-14 23:52 UTC. Generated by Docutils from reStructuredText source.