Extending the Farcry Webtop

Posted At : November 27, 2006 10:32 AM | Posted By : Mark Kruger
Related Categories: farcry

In my previous post on **Farcry Extensibility** I explained how a content "type" could be extended to support additional properties. Extending the core types only *just begins* to tap into the power of Farcry's ingenious extensibility engine. Another item that bears mentioning is the webtop. Practically anyone who has done any Coldfusion programming at all has created or used a webtop - a collection of tools that is password protected. Things like e-commerce applications, CMS systems, and email applications all come with a toolkit or webtop of some kind. Few of them make it as easy as Farcry to add new tools. It starts with the customadmin.xml file.

Creating the Customadmin.xml File

Take a look at this file located in *farcry_pliant/customadmin/* (this is a sample project used as the basis for a new install). You will see a block of XML code that is commented out. Uncomment the code and the log into the web top. You will "magically" see 2 new "tabs" along the top titled *My Custom Tab* and *Another Tab*. If you click on one of these tabs you will see a sidebar menu all setup and ready to use. Let's take a look at the structure of this file. Here's a version from CF Webtools "Dreamweaver Template Uploader" application (available **here** on the exchange).

```
<?xml version="1.0" encoding="utf-8"?>

<webtop>
    <section id="dwTemplates"
        permission="MainNavAdminTab" label="Templates">

        <subsection id="dw" permission="MainNavAdminTab"
            sidebar="custom/sidebar.cfm"
                content="inc/content_overview.html"
                            label="DW Templates">

            <menu label="Template Manager">
                <menuitem label="Add New Templates"
                    link="/admin/customadmin.cfm?module=dw/dwupload.cfm" />
                <menuitem label="Upload resources"
                    link="/admin/customadmin.cfm?module=dw/resourceupload.cfm" />
                <menuitem label="Delete Templates"
                    link="/admin/customadmin.cfm?module=dw/templatedel.cfm" />
                <menuitem label="Code XML File"
                    link="/admin/customadmin.cfm?module=dw/settings.cfm" />
                <menuitem label="Template Help"
                    link="/admin/customadmin.cfm?module=dw/help_template.cfm" />
                <menuitem label="Skin Viewer"
                    link="/admin/customadmin.cfm?module=dw/viewSkins.cfm" />
            </menu>
        </subsection>

    </section>
</webtop>
```

Let's walk through that XML file and check out each item in turn.

- *Section* - This is the tab that will appear at the top. In our case it will say

Templates. The "id" parameter is passed to the index page as the url parameter sec=dwTemplates. That's how the webtop "knows" which tab to activate and which menus to load. The "permission" tab ties into the "permissions". In this case, anyone with access to the *MainNavAdminTab* will see the "templates" tab. Finally the "label" attribute is what is displayed on the tab itself. The rest of the XML elements set up everything else under the tab.

- *Subsection* - This is the little "dropdown" box that appears at the top of the left hand sidebar. You can create and load multiple subsection into your main section. As an example, check out the "content" tab. Notice how you can switch from "dynamic content" to "content utilities" (and other choices) using the drop down box. The "sidebar" item is relative to the admin section. Leave it alone for the default behavior. The content item is the initial content to include in the main frame.
- *Menu* - This node brackets "menuitems" and it takes a label only. The label becomes the *header* for a group of links in the side bar.
- *Menuitem* - The menu item to display on the sidebar. The "link" is relative to the admin section. I'll explain more in a moment.

A couple other things of note. The *permission* attribute can be applied to a subsection, menu or menuitem for complete granular control. To see a list of permission go to the "security" tab and click on "permissions". You can add your own permission, and add it to a policy group (this is the one section that is the least friendly in Farcry). In addition, you can also add an "id" attribute to any item and a "labeltype" attribute which is set to "text" (the default) or "expression" (or "evaluate"). If set to expression then the item will be passed through the "evaluate( )" function. The "expression" method is used to pull language specific labels from a resource bundle (as in application.adminBundle[session.dmProfile.locale].content). Let's see what we have so far.

You can see our section, subsection, menu title and menu items are all set up and ready to use. Now, how do we get code to run in that space without altering the core files? Remember, the goal of true extensibility is to allow us to make changes to our project files only so that future upgrades will be painless. We want to "extend" existing functionality. There's a clue in the XML code above. Notice the link on each of the menu items. It is always something like */admin/customadmin.cfm?module=dw/dwupload.cfm*.

If you were to install the template manager and look in the "customadmin" folder you would see a subfolder called "dw". If you would look further at the farcry_core/admin/admin/customadmin.cfm file you would see that it checks permissions and then calls your file as a "cfmodule" using *projectpath*/customadmin/ as the starting point. So all that is needed to get something running in the admin is to create it inside of a customadmin. I suggest you use a folder to keep various sections separate.

Keep in mind that "enablecfoutputonly" is set to "on" so you will need to use cfoutputs around anything that is supposed to "display" in the content area. If you want to mimic the look of the other tools, use this header:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
```

```
    <title>Admin Tool title</title>
</head>
    <style type="text/css" title="default" media="screen">
        @import url(#application.url.farcry#/css/main.css);
    </style>
    <style type="text/css" title="default" media="screen">
        @import url(#application.url.farcry#/css/tabs.css);
    </style>
<body class="iframed-content">
<div id="genadmin-wrap">
<h1>Admin Tool Title</h1>
```

followed by a footer to close ranks.

```
</div>
</body>
</html>
```

Mergetype and Further Customization

It get's better. Using the "mergetype" you can even add your own menu items to existing tabs and menus. For example, if you created your own custom content type and you wanted the item to appear in the "content" section you simply have to duplicate the "content" section with the attribute *mergetype="merge"* and then add a new node. Just make sure all the id's are unique. Here's an example:

```
<section id="content"
    permission="MainNavContentTab"
        mergetype="merge"
        label="application.adminBundle[session.dmProfile.locale].content"
            labelType="evaluate">

    <subsection id="museTest"
        sidebar="content/sidebar.cfm"
        content="inc/content_overview.html"
            label="Muse Test">

            <menu id="museContent" label="Muse Content">
                <menuitem id="museNews" label="Muse News"
                    link="/admin/customadmin.cfm?module=cfmuse/musenews.cfm" />
            </menu>
    </subsection>
</section>
```

This snippet, if added to the customadmin.xml file will add a menu item called "muse test" to the *Content* tab. Accessing the tab and activating the drop down will display a menu item called *Muse News*. This is done without altering any of the existing functionality of the tab. You can even "overwrite" a section using *mergtype="replace"*.

How do I know what XML and ids to use to merge? Look in the farcry_core/config/ folder for a file called *webtop.xml*. You will see it has the same format as our customadmin.xml. Copy out the node you need, make your edits and test (make sure you don't alter webtop.xml - just customadmin.xml).

Conclusion

I have in mind to create a web service version of **CF Webstore**, an admirable e-commerce package. It occurs to me that the CF Webstore admin section can be dropped into the Farcry webtop rather easily - although a lot of work would need to be

done to make the Fusebox app work seamlessly inside the FC framework. Still, it's a project worth considering.