GENERIC SETUP

Six Feet Up Quick Reference

Best Practices

When importing items such as property sheets, make sure not to override other profile settings by setting the purge attribute to False. This will add the items listed to the property instead of resetting the property. Example:

```
<property name="metaTypesNotToList" type="lines"
purge="False">
 <element value="File"/>
 <element value="Image"/>
</property>
```

Only use the configuration that you need. When you export your site's configuration, it will include things that you don't need. For example, if you needed to change only the 'Allow anonymous to view about' property, this is what your propertiestool.xml would look like:

```
<?xml version="1.0"?>
<object name="portal_properties" meta_type="Plone
Properties Tool">
 <object name="site_properties" meta_type="Plone
Property Sheet">
  <property name="allowAnonymousViewAbout"
type="boolean">True</property>
 </object>
</object>
```

Portlets

When creating custom portlet managers, you will need to extend existing portlets to be addable to that manager:

```
<portlet extend="True" addview="portlets.Calendar">
  <for interface="my.package.interfaces.
ICustomPortletManager"/>
</portlet>
```

You can also change the title and description of the portlet with the extend attribute:

```
<portlet
  extend="True"
  title="Dates of inquisition"
  description="Nobody expects the Spanish
Inquisition!"
  addview="portlets.Calendar"/>
```

Remove a portlet definition using the 'remove' attribute so that it can no longer be added via @@manage-portlets. This does not remove any assignments:

```
<portlet remove="True" addview="portlets.Calen-
dar"/>
```

Content Generation

GenericSetup allows you to import and export content via what is called the 'structure'. There are several files that control how this works:

.objects
The .objects file contains a list of object IDs and portal_types that the structure needs to create the objects. The IDs also line up to items inside the structure folder with more information about what to create. By default all items listed will be removed and then re-added.

.preserve
The .preserve file is a list of IDs that, if present, should not be removed. This should be used if you know the profile can be run again and possibly remove your content.

.delete
The .delete file is a list of IDs that should be deleted from the site.

.properties
The .properties file typically contains information that GenericSetup will use to create the folder that it resides in. This allows the export to be represented in a nested hierarchy just as it is in the site.

Example of a .objects file taken from the Products.CMFPlone:plone profile:

```
Members,Large Plone Folder
front-page,Document
```

Example of a .properties file taken from the Products.CMFPlone:plone profile for the Members folder:

```
[DEFAULT]
description = Container for portal members' home
directories
title = Members
```

The .preserve and .delete files use the same syntax. The following would be valid to keep or delete the two objects:

```
front-page
Members
```

Other Tips

1. When installing a third party product, always make sure you have a backup.

2. Test the product installation on a local environment before applying it to production.

3. When writing profile-specific setuphandlers such as 'importVarious', make sure they only run for that profile by using context.readDataFile.

GenericSetup is a Plone/CMF product that provides a way to import and export site configuration. This allows you to make the changes in your site, then export those settings and apply them to your product.

Vocabulary

base profile
The base profile is the profile that all other profiles will extend. For users of Plone this is the 'plone' profile from the CMFPlone product.

extension profile
An extension profile is a set of configuration information that extends the base profile. Most products define at least a 'default' extension profile to set up their product.

profile version
The profile version can be set in the metadata.xml file. This tells GenericSetup what is the current version of the profile.

import steps
Import steps tell GenericSetup how to read the exported configuration for a given profile and apply it to your site.

export steps
Export steps tell GenericSetup how to export the current configuration of your site.

setup handler
A setup handler is a term given to an import step that runs some custom Python code. This is another way to create an import step.

upgrade step
An upgrade step gives you the ability to upgrade the code from one profile version to another. This is useful for one time changes that need to be made between versions.

snapshot
A snapshot can be taken of the current configuration in portal_setup. This can later be used to compare to another snapshot or profile. This can be useful when you make changes to your site and want to know how that affects your profile.

thrive

Referring to Profiles

GenericSetup refers to profiles in the following format:

```
profile-<package name>:<profile name>
```

This is the syntax that is used for dependencies in the metadata.xml. For example, if you always want to run the 'my.dependency' default profile before your profile, you would use:

```
<?xml version="1.0"?>
<metadata>
  <version>VERSION_NUMBER</version>
  <dependencies>
    <dependency>profile-my.dependency:default<dependency>
  </dependencies>
</metadata>
```

001: In GenericSetup 1.4.x the version is handled as a String, so `'1' < '14' < '2'`. More than nine upgrade steps will cause an issue. Use 3-digit version numbers, so that `'001' < '002' < '014'`.

1 or 1.0: In GenericSetup 1.5.0 and higher (used by Plone 4), this issue has been fixed and you can use a regular numbering.

Profile data is stored in a folder defined by the profile registration, in this example a folder named default inside a profiles folder. The 'name' and the directory are typically the same, but this is not required. The profile registration is typically added in the configure.zcml of your package:

```
<genericsetup:registerProfile
    name="default"
    title="My Package Profile"
    directory="profiles/default"
    description="Install profile for My Package"
    provides="Products.GenericSetup.interfaces.
EXTENSION"
    />
```

Running Profiles

The portal_setup tool is where you can directly run import profiles and perform an export. To import follow these steps:

- Login to the ZMI and go to 'portal_setup'
- Click on the 'import' tab and select the profile you want to run
- Select the specific steps you want and click 'Import Selected Steps' or click 'Import All Steps' to import everything

NOTE: The default selected profile is the 'base profile'. You should never run the 'Current base profile' as this will cause problems.

To export your site configuration you can follow these steps:

- Login to the ZMI and go to 'portal_setup'
- Select the specific steps you want and click 'Export Selected Steps' or click 'Export All Steps' to export everything

The Quick Installer can also be used to run the install and uninstall profiles. The install method defaults to running the first profile it finds. Uninstall profiles have to be wired up explicitly by the product author. Check the product's documentation for details.

Viewlets

The following examples would all be added into the viewlets.xml file.

Re-order viewlets:

```
<order manager="plone.portaltop" skinname="Plone
Default">
  <viewlet name="plone.header"/>
  <viewlet name="plone.personal_bar"/>
</order>
```

Move a viewlet using insert-before and insert-after (this will only affect the skinname that is specified, in this case 'My Custom Theme'):

```
<order manager="plone.portalheader" skinname="My
Custom Theme" based-on="Plone Default">
  <viewlet name="plone.global_sections" insert-
before="*"/>
  <viewlet name="plone.site_actions" insert-
after="plone.searchbox"/>
</order>
```

Hide a viewlet (here we hide the colophon for 'My Custom Theme'):

```
<hidden manager="plone.portalfooter" skinname="My
Custom Theme">
  <viewlet name="plone.colophon"/>
</hidden>
```

Unhide a specific viewlet using the remove attribute:

```
<hidden manager="plone.portalfooter" skinname="My
Custom Theme">
  <viewlet name="plone.colophon" remove="True"/>
</hidden>
```

Unhide all viewlets for a given manager using the purge attribute:

```
<hidden manager="plone.portalfooter" skinname="My
Custom Theme" purge="True"/>
```

Hide a viewlet for all skins:

```
<hidden manager="plone.portalfooter" skinname="*">
  <viewlet name="plone.colophon"/>
</hidden>
```

Pro Tip: In Plone prior to 4.0, using skinname="*" currently only works if the manager has already been registered in each skin (see Plone Trac ticket #7166)

Portlet Assignments

When giving a key for the context assignment, the root of the site can be referred to this way:

```
key="/"
```

Refer to the default 'news' folder in the site (NOTE: Prior to Plone 3.3.5, this required a full path like /Plone/news):

```
key="/news"
```

Delete a portlet assignment using the remove attribute:

```
<assignment
  remove="True"
  manager="plone.rightcolumn"
  category="context"
  key="/"
  type="portlets.Calendar"
  name="calendar"
  />
```

Remove all the portlet assignments for a specific manager assigned to the news object using the purge attribute:

```
<assignment
  purge="True"
  manager="plone.rightcolumn"
  category="context"
  key="/news"
  />
```

Add or move an existing portlet at the top of the column using insert-before:

```
<assignment
  insert-before="*"
  manager="plone.rightcolumn"
  category="context"
  key="/"
  type="portlets.Calendar"
  name="calendar"
  />
```

Add or move an existing portlet before the 'news' portlet:

```
<assignment
  insert-before="news"
  manager="plone.rightcolumn"
  category="context"
  key="/"
  type="portlets.Calendar"
  name="calendar"
  />
```

Pro Tip: Quickest way to find out the name of a portlet: go to @@manage-portlets and hover over the 'X'. The name for that assignment will appear in the URL.