Advanced DSpace: plugins

wiki.dspace.org/PluginManager

Three types of plugin:

- singleton plugins
- named plugins
- sequence plugins

singleton plugins

key characteristic: only one implementing class

example: site authentication

example: site authentication

```
plugin.single.org.dspace.app.webui.SiteAuthenticator = com.hp.hpl.CustomAuthenticator
```

example: site authentication

```
plugin.single.org.dspace.app.webui.SiteAuthenticator = com.hp.hpl.CustomAuthenticator
```

dspace.cfg:

```
plugin.single.org.dspace.app.webui.SiteAuthenticator = com.hp.hpl.CustomAuthenticator
```

FooServlet.java:

```
SiteAuthenticator siteAuth = (SiteAuthenticator)
        PluginManager.getSinglePlugin(SiteAuthenticator.class);
```

when to use:

when to use:

When you need a uniform mechanism to perform a given function,
but don't want to be constrained to that mechanism forever.

named plugins

key characteristic: multiple implementations available, app choses one by name

example: crosswalks

example: crosswalks

```
plugin.named.org.dspace.content.crosswalk.DisseminationCrosswalk = \
  org.dspace.content.Crosswalk.SimpleDCDisseminationCrosswalk = dc \
  org.dspace.content.Crosswalk.METSDisseminationCrosswalk = mets
```

example: crosswalks

```
plugin.named.org.dspace.content.crosswalk.DisseminationCrosswalk = \
  org.dspace.content.Crosswalk.SimpleDCDisseminationCrosswalk = dc \
  org.dspace.content.Crosswalk.METSDisseminationCrosswalk = mets
```

dspace.cfg:

```
plugin.named.org.dspace.content.crosswalk.DisseminationCrosswalk = \
  org.dspace.content.Crosswalk.SimpleDCDisseminationCrosswalk = dc \
  org.dspace.content.Crosswalk.METSDisseminationCrosswalk = mets
```

FooDisseminator.java

```
DisseminationCrosswalk xwalk = (DisseminationCrosswalk)
        PluginManager.getNamedPlugin(DisseminationCrosswalk.class, “dc”);
```

note:

named plugins must have names (keys) that are unique for the interface

note:

names may contain any characters other than
',' and '=', but alphanumeric is preferred

when to use:

when to use:

When you need multiple implementations to be available / supported throughout the application, leaving the implementing classes to choose which option to use.

sequence plugins

key characteristic: “stack of singletons”

example: stackable authentication

example: stackable authentication

```
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \
  org.dspace.authenticate.X509Authentication, \
  org.dspace.authenticate.PasswordAuthentication
```

example: stackable authentication

```
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \
  org.dspace.authenticate.X509Authentication, \
  org.dspace.authenticate.PasswordAuthentication
```

dspace.cfg:

```
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \
  org.dspace.authenticate.X509Authentication, \
  org.dspace.authenticate.PasswordAuthentication
```

AuthenticationManager.java:

```
private static AuthenticationMethod stack[] = (AuthenticationMethod[])
        PluginManager.getPluginSequence(AuthenticationMethod.class);
```

when to use:

when to use:

When you want to pass the responsibility for processing through a stack of implementing classes.

wiki.dspace.org/Category:HOWTO
wiki.dspace.org/Guide_to_Developing_with_DSpace

dspace-devel@lists.sourceforge.net