# Advanced DSpace

## plugins, configurable submission, & core API

James Rutherford, DSUG Rome 2007

# Part 1: plugins

Part 1: 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.

Part 2: DSpace API

wiki.dspace.org/static_files/7/7d/DevelopersDocumentation.pdf
wiki.dspace.org/DAO_Prototype (needs updating)

# Section 1: 1.4.x & 1.5

quick overview: creating a hierarchy

```
Community community = Community.create(null, context);
community.setMetadata(“short_description”, “top-level community”);
community.update();
```

parent community (optional)



```
Community community = Community.create(null, context);
community.setMetadata(“short_description”, “top-level community”);
community.update();
```

```
Community community = Community.create(null, context);
community.setMetadata(“short_description”, “top-level community”);
community.update();

Collection collection = community.createCollection();
collection.setMetadata(“short_description”, “collection foo”);
collection.update();
```

checks authorization, creates Collection, & establishes parent / child relationship



```
Collection collection = community.createCollection();
collection.setMetadata(“short_description”, “collection foo”);
collection.update();
```

```
Community community = Community.create(null, context);
community.setMetadata(“short_description”, “top-level community”);
community.update();

Collection collection = community.createCollection();
collection.setMetadata(“short_description”, “collection foo”);
collection.update();

Item item = Item.create(context);
collection.addItem(item);
```

```
Item item = Item.create(context);
collection.addItem(item);
```



performs authorization checks and establishes the parent / child relationship

or

destination Collection

```
WorkspaceItem wsi = WorkspaceItem.create(context, collection, false);
Item item = wsi.getItem();
```

Item will start as a copy of the Collection's template

```
Community community = Community.create(null, context);
community.setMetadata(“short_description”, “top-level community”);
community.update();

Collection collection = community.createCollection();
collection.setMetadata(“short_description”, “collection foo”);
collection.update();

Item item = Item.create(context);
collection.addItem(item);

-- or --

WorkspaceItem wsi = WorkspaceItem.create(context, collection, false);
Item item = wsi.getItem();
```

Section 2: 1.6+*

* core API for 1.6+ releases still needs to be finalised

comparison with existing API: creating a hierarchy

```
CommunityDAO communityDAO = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO = ItemDAOFactory.getInstance(context);
```

```
CommunityDAO communityDAO   = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO             = ItemDAOFactory.getInstance(context);
```

implementation defined in configuration

```
CommunityDAO communityDAO = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO = ItemDAOFactory.getInstance(context);

Community community = communityDAO.create();
community.setMetadata(“short_description”, “top-level community”);
communityDAO.update(community);
```

```
CommunityDAO communityDAO = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO = ItemDAOFactory.getInstance(context);

Community community = communityDAO.create();
community.setMetadata(“short_description”, “top-level community”);
communityDAO.update(community);

Collection collection = collectionDAO.create();
collection.setMetadata(“short_description”, “collection foo”);
collectionDAO.update(collection);
```

```
CommunityDAO communityDAO = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO = ItemDAOFactory.getInstance(context);

Community community = communityDAO.create();
community.setMetadata(“short_description”, “top-level community”);
communityDAO.update(community);

Collection collection = collectionDAO.create();
collection.setMetadata(“short_description”, “collection foo”);
collectionDAO.update(collection);

Item item = ItemDAO.create();
```

```
CommunityDAO communityDAO = CommunityDAOFactory.getInstance(context);
CollectionDAO collectionDAO = CollectionDAOFactory.getInstance(context);
ItemDAO itemDAO = ItemDAOFactory.getInstance(context);

Community community = communityDAO.create();
community.setMetadata(“short_description”, “top-level community”);
communityDAO.update(community);

Collection collection = collectionDAO.create();
collection.setMetadata(“short_description”, “collection foo”);
collectionDAO.update(collection);

Item item = ItemDAO.create();

communityDAO.link(community, collection);
collectionDAO.link(collection, item);
```

# Section 2a: other changes coming in 1.6

## Section 2a: other changes coming in 1.6

- `SQLExceptions`

## Section 2a: other changes coming in 1.6

- ~~`SQLExceptions`~~

## Section 2a: other changes coming in 1.6

- ~~SQLExceptions~~
- DSpaceObject.getHandle()

## Section 2a: other changes coming in 1.6

- ~~SQLExceptions~~
- DSpaceObject.~~getHandle()~~

## Section 2a: other changes coming in 1.6

- ~~SQLExceptions~~
- DSpaceObject.~~getHandle()~~
- UUIDs

## Section 2a: other changes coming in 1.6

- ~~SQLExceptions~~
- DSpaceObject.~~getHandle()~~
- UUIDs
- ObjectIdentifier

## Section 2a: other changes coming in 1.6

- ~~SQLExceptions~~
- DSpaceObject.~~getHandle()~~
- UUIDs
- ObjectIdentifier
- item.update() --> itemDAO.update(item)

# Section 3: org.dspace.core.Context

example 1: basic use

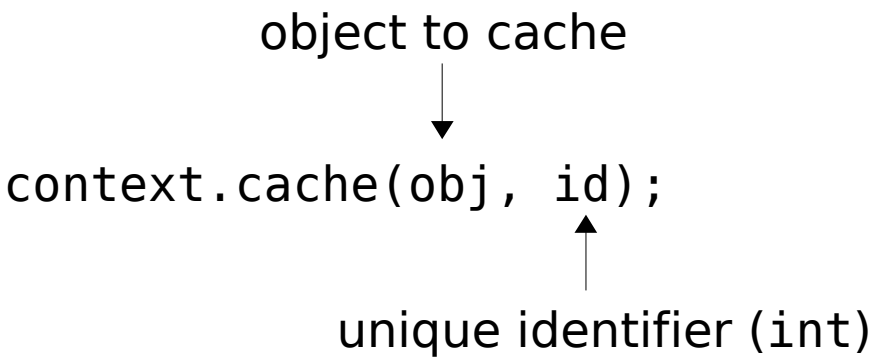
```
Context c = null;

try
{
    c = new Context();
    ...
    c.commit();
    ...
    c.complete();
}
catch (SQLException sqle)
{
    c.abort();
    c = null;

    log.warn(“context aborted”);
    System.err.println(“context aborted”, sqle);
}
```

# example 2: cache

```
context.cache(obj, id);
```


object to cache
context.cache(obj, id);
unique identifier (int)

```
context.cache(obj, id);

Foo obj = (Foo) context.fromCache(Foo.class, id);
```

```
Foo obj = (Foo) context.fromCache(Foo.class, id);
```

once an object is cached, we can retrieve
it from memory if we know the id

```
context.cache(obj, id);

Foo obj = (Foo) context.fromCache(Foo.class, id);

context.removeCached(obj, id);
```

```
context.removeCached(obj, id);
```



clean up after yourself :)

```
context.cache(obj, id);

Foo obj = (Foo) context.fromCache(Foo.class, id);

context.removeCached(obj, id);

context.clearCache();
```

```
context.clearCache();
```



clean up after others :)

```
context.cache(obj, id);

Foo obj = (Foo) context.fromCache(Foo.class, id);

context.removeCached(obj, id);

context.clearCache();
```

cache tips:

- always inspect cache before reading from data store
- don't cache unnecessarily

thanks

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

dspace-devel@lists.sourceforge.net