DSpace DAO Prototype

Note: As of August 2009, the code this refers to is located at http://scm.dspace.org/trac/dspace/browser/dspace/branches/dspace-dao-prototype, but is unlikely to be released from there. A version called "1.6" that is based on the /dspace_1_5_x branch will be released. – Stuart Lewis, August 20, 2009

Note: As of April 2009, the code this refers to is located at the repository /trunk, but is unlikely to be released from there. A version called "1.6" that is based on the /dspace_1_5_x branch will be released, and may incorporate some of these changes. – Bradley McLean, April 13, 2009

---

*__Author__: James Rutherford
*__Version__: 2007-11-05

- See also: DAO Prototype, PersistentIdentifiers (both slightly out of date), Google+Summer+of+Code+2007 Versioning

NOTE: *this page is a work in progress.*

This page exists to document the new (as of Q4 2007) features introduced into DSpace post 1.5. This is intended for developers, mostly as a guide for how to migrate from coding against the 'old' API to working with the new features as they exist in DSpace+1.6+. The three main features that I will outline in this page are DAOs, identifiers, and versioning. Note that these aren't the only new things that will appear in 1.6.

TODO

I'll start by outlining what still needs to be fixed / refactored / tested more thoroughly, so you have an idea of what's missing before you start.

Consistency

There are several areas that need to be aligned to make the application more consistent. A lot of work has already been done here, but there is much left to do:

- logging
  - level, detail, etc
- Events
- authorization
- use of `org.dspace.core.Context` cache

API

The API has already changed considerably (see below), but there are still a few things that I need to straighten out:

- use of `org.dspace.content.uri.ObjectIdentifier` instead of `int` or `String` "identifiers".
- pre- and post-hooks for all DAO operations (or at least the basic CRUD).
- resolve API conflicts such as `item.getBundles()` vs. `bundleDAO.getBundles(item)` (the former asks the in-memory `Item`, the latter queries the data store – results will not necessarily be the same).

URLs

To ease migration away from the dependency on Handles, the 'new' URL form is basically the same as it was (`site_url/resource/identifier/extras`). While this basically, works, there are bugs waiting to creep in (mostly relating to identifiers that contain unescaped slashes), and it may be worth moving towards a more parameterized URL scheme.

Constructors

Classes that have corresponding DAOs (`Item` etc) typically had package-private constructors which, among other things, prevented them from being subclassed (which was irritating); these constructors (usually) took a `TableRow` and a `Context`. In order to maintain some degree of consistency with the old API, such constructors now take an `int` ID, and a `Context`. It is important to note that these constructors are only really for use by the DAOs – all creates and retrieves go via DAOs so there should never be a need to directly instantiate one of these objects directly.

In the near future, I will probably remove these constructors and replace them with constructors that take no arguments (the `Context` is only required to support deprecated methods anyway) to avoid confusion.

Things that are broken

- `org.dspace.app.itemimport.ItemImport` is broken (because of the previous dependency on Handles as identifiers). I'm in the process of fixing this, but for now it just doesn't work.

DAOs

Example: `ItemDAO`

Sample code to follow.

`org.dspace.content.proxy`

This section will provide an overview of the new `ItemProxy` class.

Identifiers

It used to be the case that DSpace used a mix of `int` and `String` representations of Handles to identify objects. As much as possible, I've swapped all of these out in favour of using the new `ObjectIdentifier` class. The idea behind this identifier was driven by one of the reasons for using DAOs: n