Package org.dspace.content

Provides an API for reading and manipulating content in the DSpace system.

See:

Description

Interface Summary	
InProgressSubmission	Interface for manipulating in-progress submissions, without having to know at which stage of submission they are (in workspace or workflow system)

Class Summary	
Bitstream	Class representing bitstreams stored in the DSpace system.
BitstreamFormat	Class representing a particular bitstream format.
Bundle	Class representing bundles of bitstreams stored in the DSpace system
Collection	Class representing a collection.
Community	Class representing a community
CommunityGroup	Class representing a community group.
DCDate	Dublin Core date utility class
DCLanguage	Utility class for dealing with languages
DCPersonName	DSpace person name utility class
DCSeriesNumber	Series and report number, as stored in relation.ispartofseries
DCValue	**Deprecated.**
DSpaceObject	Abstract base class for DSpace objects
EtdUnit	Class representing an ETD department as seen in the Proquest metadata element /DISS_submission/DISS_description/DISS_institution /DISS_inst_contact
FormatIdentifier	This class handles the recognition of bitstream formats, using the format registry in the database.
InstallItem	Support to install an Item in the archive.

Item	Class representing an item in DSpace.
ItemComparator	Compare two Items by their DCValues.
ItemIterator	Specialized iterator for DSpace Items.
LicenseUtils	Utility class to manage generation and storing of the license text that the submitter has to grant/granted for archiving the item
MetadataField	DSpace object that represents a metadata field, which is defined by a combination of schema, element, and qualifier.
MetadataSchema	Class representing a schema in DSpace.
MetadataValue	Database access class representing a Dublin Core metadata value.
Site	Represents the root of the DSpace Archive.
SupervisedItem	Class to handle WorkspaceItems which are being supervised.
Thumbnail	Wrapper class for bitstreams with Thumbnails associated with them for convenience in the browse system
WorkspaceItem	Class representing an item in the process of being submitted by a user

Exception Summary	
NonUniqueMetadataException	An exception that gets thrown when a metadata field cannot be created or saved due to an existing field with an identical element and qualifier.

Package org.dspace.content Description

Provides an API for reading and manipulating content in the DSpace system.

The DSpace Data Model

Data in DSpace is stored in the model below. Multiple inclusion is permitted at every level; the documentation for each class describes the system's behaviour for coping with this.

Community	Communities correspond to organisational units within an institution.
Collection	Collections are groupings of related content. Each collection may have an associated *workflow*; this is the review process that submissions go through before being included in the archive.
Item	Items are the basic archival units. An item corresponds to a single logical piece of content and associated metadata.

Bundle	Bundles are groupings of Bitstreams that make no sense in isolation; for example, the files making up an HTML document would all go in one Bundle. A PDF version of the same Item, or a dataset stored with the Item, would go in a separate Bundle.
Bitstream	Bitstreams are sequences of bits, typically files, that make up the raw content of Items.

Additionally, each Bitstream is associated with one **Bitstream Format**; this describes information about the format and encoding of the Bitstream, including a name (for example "Adobe PDF"), a MIME type and a support level.

Submissions are created as **Workspace Items**. A Workspace Item is an Item in progress. Once item assembly is complete, one of two things may happen:

- If the Collection being submitted to has an associated workflow, it is started. At this point the Workspace Item becomes a **Workflow Item**.
- If the Collection has no associated workflow, the Workspace Item is removed and the assembled Item is included in the Collection.

Workspace Items and Workflow Items may both be manipulated as In Progress Submissions.

Using the Content Management API

The general paradigm for using DSpace is to create a *Context*; this is akin to opening a connection to a database (which, coincidentally, is one of the things that happens.)

The classes in this package are then used to create in-memory snapshots that represent the corresponding logical objects stored in the system. When the reading or manipulating is done, the Context may either be *aborted*, in which case any changes made are discarded, or *completed*, in which case any changes made are committed to main DSpace storage.

If any error occurs if you are making changes, you should *abort* the current context, since the in-memory snapshots might be in an inconsistent state.

Typically, when changing a particular object in the system, the changes will not be written to main DSpace storage unless `update` is called on the object prior to Context completion. Where this is not the case, it is stated in the method documentation.

Instances of the classes in this package are tied to that Context; when the Context has been finished with the objects essentially become invalid.

An example use of the Content Management API is shown below:

```
try
{
    // Create a DSpace context
    context = new org.dspace.core.Context();

    // Set the current user
    context.setCurrentUser(authenticatedUser)
```

```
    // Create my new collection
    Collection c = Collection.create(context);
    c.setMetadata("name", "My New Collection");
    c.update();   // Updates the metadata within the context

    // Find an item
    item = Item.find(context, 1234);

    // Remove it from its old collections
    Collection[] colls = item.getCollections();
    colls[0].removeItem(item);

    // Add it to my new collection
    c.addItem(item);

    // All went well; complete the context so changes are written
    context.complete();
}
catch (SQLException se)
{
    // Something went wrong with the database; abort the context so
    // no changes are written
    context.abort();
}
catch (AuthorizeException ae)
{
    // authenticatedUser does not have permission to perform one of the
    // above actions, so no changes should be made at all.
    context.abort();
}

// The context will have been completed or aborted here, so it may
// no longer be used, nor any objects that were created with it (e.g. item)
```

See Also:

org.dspace.authorize, Context
