Islandora BagIt

Overview

BagIt is a specification for packaging content and metadata about that content into a format that can be shared between applications. This module provides a "Create Bag" option that allows the packaging of the datastreams in Islandora objects. This module is a fork of BagIt.

Dependencies

This module requires the following modules/libraries:

- Islandora
- Tuque
- Scholars' Lab BagItPHP library
- Archive_Tar
- Libraries

Downloads

Release Notes and Downloads

Installation

To install the Islandora BagIt module:

1. Install Archive_Tar. This package is required by PEAR so if you have PEAR installed on your system, you won't need to install Archive_Tar separately.
2. Install the Libraries API contrib module.
3. Unzip this module into your site's modules directory as you would any other contrib module.
4. Install the BagItPHP library by entering your site's sites/all/libraries directory and issuing the following command:

   ```
   git clone git://github.com/scholarslab/BagItPHP.git
   ```
5. Enable the Libraries and Islandora BagIt modules like you would any other contrib modules.

Configuration

Set the location for the BagIt library, temporary directory for unserialized Bags, output directory for serialized Bags, Bag name prefix, select Collection batch type, compression type, object plugins, collection plugins, and set Bag Metadata in Administration » Islandora » Islandora BagIt (admin/islandora/bagit).

blocked URL

Extending and customizing the BagIt module

Islandora BagIt uses two diffent types of plugins, object plugins and collection plugins.

Object plugins add files to an Islandora object's Bag. Object plugins 1) define the source path for datastream files and their destination paths in the Bag relative to the Bag's 'data' directory, and 2) optionally, rename datastream files or create new files for inclusion in the Bag.

The module comes with five object plugins: one that copies all the datastreams in an Islandora object to the top level of the Bag's 'data' directory, one that illustrates how to create an arbitrary file to add to the Bag, one that includes the FOXML for an Islandora object, one that creates an Archivematica (http://archivematica.org) 'transfer' Bag, and one that includes a PREMIS preservation metadata file generated by the Islandora PREMIS module (https://github.com/ruebot/islandora_premis).

Multiple object plugins may be activated, but they all add files to the same Bag. This means you can write a plugin that creates one specific file (for example) and activate it or deactivate it as needed. Plugins are fired in the order in which their filenames sort.

If you have requirements not covered by the supplied object plugins, you can use the plugins described above to base your own on. If you write your own plugins, follow these guidelines:

1. Begin plugin filenames with 'plugin_object_' and end in '.inc'.
2. Every plugin has a function that is named the same as the filename prepended with 'islandora_bagit_', ending in '_init().' All init functions take $islandora_object and $tmp_ds_directory parameters.
3. Plugins must complete all file writing and copying tasks before they return the file's source and destination paths to the Bagit module.
4. Plugins return FALSE if there is an error in copying or writing files.

Collection plugins only apply to Bags that contain all the objects in a collection. Whereas object plugins determine what goes in each object-level Bag, collection plugins determine how all the objects that are in a collection-level Bag are arranged in the Bag's 'data' directory. Also, you can only activate one collection plugin at a time, unlike object plugins.

The requirements for collection plugins are the same as those for object plugins except for the requirement a) above: collection plugin filenames being with 'plugin_collection_' instead of 'plugin_object_'.

The module comes with two collection plugins, one that creates a subdirectory for each object in the Bag's 'data' directory, and one that creates an 'odd' and 'even' subdirectory in the Bag's 'data' directory, and then organizes object-level Bags within those two subdirectories. The odd/even plugin is intended to illustrate alternative ways to organize objects within a collection Bag.

Modifying a Bag from your own modules

This module provides a drupal_alter() hook, which allows other modules to use hook_islandora_bagit_alter($bag, $islandora_object). Your module can modify the current Bag using any of the methods provided by the BagItPHP library. Each implementation of this hook must take $bag and $islandora_object as parameters; $islandora_object is provided so you can access properties of the object in your module easily. A typical implementation looks like this:

```
/**
 * Implementation of hook drupal_alter().
 *
 * @param object $bag
 *   A BagIt object instantiated in the BagIt module.
 *
 * @param object $islandora_object
 *  The current $islandora_object.
 */
function mymodule_islandora_bagit_alter($bag, $islandora_object) {
  // Add some custom metadata to bag-info.txt.
  $bag->bagInfoData('Some-Arbitrary-Field', 'Foo bar baz');
  // Add a file that is not managed by a plugin. Note: extra files
  // should be added by plugins if possible, since files that are
  // added in drupal_alter() hooks are not counted in Payload-Oxum
  // values generated by the Islandora BagIt module.
  $bag->addFile('/path/to/file.txt', 'myfile.txt');
  // Update the Bag (this is required).
  $bag->update();
}
```

Note that implementations of hook_islandora_bagit_alter() must call $bag->update() themselves, typically at the very end of the function.

Modifying a batch from your own modules

If you are running a collection-level batch to create a Bag for every object in a collection, or to create a single Bag containing all objects in a collection, you can define filters on which objects get included using hook_islandora_bagit_filter_batch(). If you want an object to be excluded from the batch, have your instance of this hook return TRUE; if you want an object to be included, return FALSE (or don't issue an explicit 'return' at all). This example instance excludes objects with either of two specific PIDs:

```
/**
 * Implementation of hook_islandora_bagit_filter_batch().
 *
 * @param string $pid
 *   A BagIt object instantiated in the BagIt module.
 *
 * @return bool
 */
function mymodule_islandora_bagit_filter_batch($pid) {
  if ($pid == 'islandora:87' || $pid == 'islandora:91') {
    return TRUE;
  }
}
```

If you want to test other attributes of the object, you need to use Islandora's islandora_object_load($pid) function to load the object so you can access the attributes.

Post-Bag-creation hook

Islandora BagIt provides an additional hook, islandora_bagit_post_create, that allows other modules to get notifications that a Bag has just been created. A basic implementation is:

```
/**
 * Implements hook_islandora_bagit_post_create().
 *
 * @param string $pid
 *   The PID of the Islandora object that the Bag was just created for.
 *
 * @param string $bag_path
 *  The path to the Bag, relative to the Drupal installation directory.
 */
function mymodule_islandora_bagit_post_create($pid, $bag_path) {
  // Do something interesting.
}
```

This hook can be used to send notification emails after a Bag has been created, to add the Bag to a queue for further processing, or to copy the Bag to a different server.

Drush integration

Bags can be created for individual Islandora objects or for all objects in a given collection using Drush:

```
drush --user=UID create-islandora-bag [object|collection] PID
```

where UID is the user ID or user name of the fedoraAdmin user (or equivalent), 'object' or 'collection' indicates whether you want to create a Bag for a single object or a Bag for every member of a collection, and PID is the PID of the Islandora object or collection.

Permissions and security

This module is intended for users who have a fairly high level of permissions on a Drupal site. Because the goal is to package up all or some of the datastreams in an Islandora object, users who can create and download Bags should have access to those datastreams. However, the module does check the current users' access to a datastream before adding it to the Bag.

Known Issues

Fedora 3.8.0 fails to generate FOXML files requested using the 'archive' context (JIRA ticket). Earlier versions may succeed on exporting 'archive' FOXML files if the resulting FOXML is smaller than approximately 200 MB, but fail on larger files. The Islandora BagIt module triggers this set of errors if 'archive' FOXML files are generated from within one of its plugins (JIRA ticket). Until this issue is resolved in Fedora, users of the Islandora BagIt module should not use plugins that generate 'archive' FOXML, including plugin_object_foxml.inc distributed with versions of Islandora BagIt prior to 7.x-1.5. The other FOXML export contexts, 'public' and 'migrate' (documentation), can be used safely.

Some bags do not finish properly even with PHP CLI's php.ini set to `max_input_time = -1` (JIRA ticket). For better performance the CLI's php.ini should have the lines `max_execution_time = 0` and `man_input_time = -1`, and well as including `Timeout 86400` in Apache2's apache2.conf.