Infrastructure - Task #3107

Task # 3074 (Closed): Phase one implementaion for morpho to connect dataone services

Need a mechanism to keep track the identifer's history (revision)

2012-07-30 21:55 - Jing Tao

Status:	Closed	**Start date:**	2012-07-30
Priority:	Normal	**Due date:**	
Assignee:	Jing Tao	**% Done:**	100%
Category:	Morpho	**Estimated time:**	0.00 hour
Target version:	2013.10-Block.2.1		
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

Currently morpho use the identifier itself to keep track its history. For example. foo.300.2 is a newer version of foo.300.1.

The new morpho will change the policy. Instead, we should look up the system metadata associated with an object to find out its history or newer version.

System metadata has two elements named obsoletes/obsoletedBy

Here is an example:
System metadata for object 100 is: 200
System metadata for object 200 is: 300
System metadata for object 300 doesn't have the "obsoletes" element.

The newest version is object 100, the oldest version is 300. 200 is the middle.

Morpho has two situations needing to know the version history.
1. Building the query result set. The query result set only shows the newest version. For the network search, it is easy since we only get the newest version. But the local search will return every document matching the query. So we needs to trim the old version documents. Also, we need to compare the versions of the network copy and local copy. If they are same, the document's location status will show both. If the network copy has newer version, the location status will show network. If the local copy has newer version, the location status will show local.

    1. Menu item for "Open the old version".

We may have two options to keep track the revisions:
1. When we needs the information, we search the system metadata folder to get it. I can imagine the response time will be long.
2. When morpho starts up, morpho will go through the metadata folder to collect the revision information and keep them in an object(i.g., a list of list). When we needs the information, just call the object directly. The response time will be short. However, the morpho start-up time may be long if the user has a big local data set. The solution maybe is to cache the object to a csv file. So only the first time to start morpho will take a long time.

History

#1 - 2012-10-03 15:59 - Ben Leinfelder

- Assignee set to Jing Tao

I think Morpho should keep track of revision history using it's own mechanism and should only be inspired by the DataONE approach. This way we can implement the revision tracking feature using our existing Metacat docids (that encode the revision history in their value) and then moving to the opaque identifiers will be simple when we add support for DataONE.

The general utility is something like:
List RevisionManager.getInstance().getAllRevisions(String identifier)
String RevisionManager.getInstance().getObsoletes(String identifier)
String RevisionManager.getInstance().getObsoletedBy(String identifier)
RevisionManager.getInstance().setObsoletes(String old, String new)
RevisionManager.getInstance().setObsoletedBy(String old, String new)

I'm not sure how best to serialize this information for the actual implementation. Simplest might be two properties files (obsoletes.properties and obsoletedBy.properties) that have key=value pairs of identifiers. Would be a bit of processing (lots of properties.get() calls!) to get the complete list of revisions for any given identifier, but would be very simple storage mechanism.

#2 - 2012-10-11 15:33 - Dave Vieglais

- Target version changed from Sprint-2012.37-Block.5.3 to Sprint-2012.41-Block.6.1

#3 - 2012-10-14 15:09 - Ben Leinfelder

- % Done changed from 0 to 10

- Category set to Morpho

We have a plan for the implementation (xml+apache configuration lib) that will be generalized for use with Metacat docids and also DataONE systemmetadata versioning.

Matt, Jing, and Ben decide that the revision chain will be different between local and remote locations. For example, an object can be created (r1) and edited locally (r2, r3) before being saved to the network for the first time (r3, no revision history). So the RevisionManager will be per-store (local vs remote).

#4 - 2012-10-24 18:20 - Ben Leinfelder

- Target version changed from Sprint-2012.41-Block.6.1 to Sprint-2012.44-Block.6.2

#5 - 2012-12-12 16:51 - Chris Jones

- Target version changed from Sprint-2012.44-Block.6.2 to Sprint-2012.50-Block.6.4

#6 - 2013-03-01 18:33 - Ben Leinfelder

- Target version changed from Sprint-2012.50-Block.6.4 to 2013.10-Block.2.1

#7 - 2013-03-02 05:30 - Ben Leinfelder

- Parent task changed from #3075 to #3635

#8 - 2013-03-02 05:32 - Ben Leinfelder

- Parent task changed from #3635 to #3074

#9 - 2013-03-25 22:54 - Ben Leinfelder

- translation missing: en.field_remaining_hours set to 0.0

- Status changed from New to Closed

This has been implemented in Morpho. There is some ongoing discussion in the team about how to handle network vs local revision history, but all that traffic is in Bugzilla and the soon to be ecoinformatics redmine instance. http://bugzilla.ecoinformatics.org/show_bug.cgi?id=5736