Arvados - Bug #5145

[Workbench] Combining collections should separate files with identical names, not concatenate them

02/05/2015 09:38 PM - Bryan Cosca

Status:	Resolved	**Start date:**	03/05/2015
Priority:	Normal	**Due date:**	
Assigned To:	Radhika Chippada	**% Done:**	100%
Category:	Workbench	**Estimated time:**	0.00 hour
Target version:	2015-03-11 sprint		

Description

collection uuid
foo.txt hash 123

add foo.txt hash 234 to collection uuid

Currently this fails, is there a way to change this in a way that the file gets added but maybe the name gets changed? This came up in a customer meeting.

Subtasks:

Task # 5386: Review branch: 5145-combine-collections-repeated-filenames	**Resolved**

Associated revisions

Revision 00c9cd4e - 03/11/2015 06:44 PM - Radhika Chippada

closes #5145
Merge branch '5145-combine-collections-repeated-filenames'

Revision 593e8c87 - 03/11/2015 09:20 PM - Radhika Chippada

refs #5145
Merge branch '5145-combine-collections-repeated-filenames'

Revision 20f3649b - 03/12/2015 01:36 PM - Peter Amstutz

Fix arv-normalize regression. refs #5145 refs #4823

History

#1 - 02/06/2015 03:53 PM - Brett Smith

Bryan,

Can you please explain the original bug/failure in more detail? Specifically, I'd like to know what tool you used to try to add the second file to the existing collection. The Workbench uploader has code to dynamically rename new files.

#2 - 02/06/2015 03:53 PM - Brett Smith

- Target version set to Bug Triage

#3 - 02/18/2015 02:43 PM - Bryan Cosca

so in here: qr1hi-j7d0g-0x4b08b68zanwxh
I have new collection and new collection (2)
I have foo.txt in each of them (one has text bar and the other foo)
I then combine the collections together and I got the collection qr1hi-4zz18-020vo4bxoc3vo1b which shows foo.txt concatenated with both of the original foo.txt (barfoo)
I think these should be two separate items.

#4 - 02/20/2015 07:52 PM - Brett Smith

- Subject changed from Adding a file to a collection with the same name as a file in a collection to [Workbench] Combining collections should separate files with identical names, not concatenate them

- Category set to Workbench

#5 - 03/03/2015 06:24 PM - Radhika Chippada

There are two distinct scenarios here.

- Scenario 1: Two files from two different directories of a Collection are combined. The files have same name and same content. Currently, this results in error: the new collection will have one file with that name whose contents are file1 contents + file2 contents (basically two copies of the same content). I think in this case, we can include just one copy of the file.

- Scenario 2: Two files from two different directories of a Collection are combined. The files have same name but different content. In this case, this results in error: the new collection will have one file with that name whose contents are file1 contents + file2 contents. How do we want resolve this scenario?

#6 - 03/03/2015 06:48 PM - Tom Clegg

Radhika Chippada wrote:

> Scenario 1: Two files from two different directories of a Collection are combined. The files have same name and same content. Currently, this results in error: the new collection will have one file with that name whose contents are file1 contents + file2 contents (basically two copies of the same content). I think in this case, we can include just one copy of the file.

Is there a known/common use case for this? I'm hesitant to add special behavior for this situation:

1. Sometimes it's obvious that file1 and file2 have the same content, but sometimes it isn't: different chunking can encode the same content using different block segments/hashes. It's hard for a Workbench user to see this distinction, so this behavior could end up being perceived as "unpredictably, duplicates are sometimes skipped and sometimes not".
2. Even if the behavior is consistent, it still seems a bit mysterious: if I combine two collections with 100 files each, I get a collection with 199 files. Now I have to investigate why one file went missing. This could be addressed by flashing a notice "We skipped some duplicate files".
3. What if I really wanted 200 files? Circumventing this feature could get awkward. OTOH, it's easy to hit a "delete file" button to remove duplicates by myself if that's what I want. (In that respect, a "delete file" feature would be a higher-priority usability improvement than automatic de-duplication.)

If we do have a de-duplication feature, I think it should be explicit: "You've just added multiple identical copies of some files to your collection. Want me to remove the redundant copies for you?" (But I don't think this feature should be bundled into the current bugfix.)

> Scenario 2: Two files from two different directories of a Collection are combined. The files have same name but different content. In this case, this results in error: the new collection will have one file with that name whose contents are file1 contents + file2 contents. How do we want resolve this scenario?

I think we should do what the web uploader does in this situation: When adding "/foo.txt" to a collection that already has "/foo.txt", call the new file "/foo (X).txt" where X is the smallest positive integer that doesn't conflict with an existing file.

For purposes of this bugfix, I think scenario 1 and 2 can be treated exactly the same. ("Adding duplicate files results in duplicate files" might not be what the user wants in some cases, but I wouldn't call it a bug...)

#7 - 03/04/2015 04:06 PM - Radhika Chippada

- Status changed from New to In Progress

- Assigned To set to Radhika Chippada

- Target version changed from Bug Triage to 2015-03-11 sprint

#8 - 03/05/2015 03:35 PM - Radhika Chippada

- Updated combine action to handle duplicate names by appending a number to the file name being repeated.

- Combined collection may see duplicate names when same file names from different directories of a collection are being combined.

- It is also possible that the collections being combined have directories with same file names with same or different contents. When file names are repeated within a directory name, their names will only conflict with other files within that that same directory / stream. Other files in other streams will not be considered as duplicate names.

- Tested manually by combining collections or files with same names and different contents. Verified by doing keep get on the resulting collection.

- Added controller tests.

#9 - 03/09/2015 06:34 PM - Peter Amstutz

```
normalized = normalized.gsub(manifest_file) {|s| uniq_file}
```

This needs to match more specifically, otherwise this could happen:

```
". abc+5 0:5:abc".gsub("abc") { |s| "xyz" }
=> ". xyz+5 0:5:xyz"
```

You need something like this:

```
normalized = normalized.gsub(/\d+:\d+:(#{Regexp.quote manifest_file})/) {|s| uniq_file}
```

```
 part_match = /\d*:\d*:(.*)/.match(part)
```

Should be

```
part_match = /\d+:\d+:(\S+)/.match(part)
```

and the previous comment for gsub applies here as well:

```
adjusted_parts << (part.gsub(part_match[1]) {|s| uniq_file})
```

You should normalize mt before processing it to ensure that there are no filenames with '/' in them.

#10 - 03/09/2015 09:04 PM - Radhika Chippada

- normalized = normalized.gsub(/\d+:\d+:(#{Regexp.quote manifest_file})/) {|s| uniq_file}

This is resulting in removing the \d+:\d+: portion. Instead I split the string on file name index and applied gsub on the last part, which is the file name being replaced.

- part_match = /\d+:\d+:(\S+)/.match(part)

Done

- and the previous comment for gsub applies here as well: (adjusted_parts << (part.gsub(part_match[1]) {|s| uniq_file}))

This is not required since I am doing gsub on the filename part alone and hence there are no other such occurrences

- You should normalize mt before processing it to ensure that there are no filenames with '/' in them.

Done

- Also needed to update one collection fixture to have correct format (it was missing the locator)

#11 - 03/10/2015 12:55 PM - Peter Amstutz

Radhika Chippada wrote:

> - normalized = normalized.gsub(/\d+:\d+:(#{Regexp.quote manifest_file})/) {|s| uniq_file}
>
> This is resulting in removing the \d+:\d+: portion. Instead I split the string on file name index and applied gsub on the last part, which is the file name being replaced.

You're right, I was wrong about it just matching the subexpression, here's the right way to do it:

```
normalized.gsub(/(\d+:\d+:)(#{Regexp.quote manifest_file})/) {|s| "#{$1}#{uniq_file}" }
```

This the first subexpression goes into the variable $1 and uses that to build a new string using the new filename.

> - part_match = /\d+:\d+:(\S+)/.match(part)
> Done

I still see

```
part_match = /\d*:\d*:(\S+)/.match(part)
```

\d* is wrong since that will match zero-length strings.

> - and the previous comment for gsub applies here as well: (adjusted_parts << (part.gsub(part_match[1]) {|s| uniq_file}))
>
> This is not required since I am doing gsub on the filename part alone and hence there are no other such occurrences

I think you want do something like:

```
part_match = /(\d+:\d+:)(\S+)/.match(part)
```

```
if part_match
  uniq_file = derive_unique_filename(part_match[2], manifest_files)
  adjusted_parts << "#{part_match[1]}#{uniq_file}"
else
  adjusted_parts << part
end
```

#12 - 03/11/2015 04:20 PM - Radhika Chippada

Peter, thanks for the regexp lesson. Addressed all three items. Thanks.

#13 - 03/11/2015 06:43 PM - Peter Amstutz

b0cf100 looks good to me.

#14 - 03/11/2015 07:00 PM - Radhika Chippada

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:00c9cd4ecab3683d95118ab9d68310ac5069e9f6.