TYPO3.Flow - Task # 36800

Status:	Accepted	**Priority:**	Should have
Author:	Robert Lemke	**Category:**	Resource
Created:	2012-05-03	**Assigned To:**	Robert Lemke
Updated:	2013-08-02	**Due date:**	
Sprint:			
PHP Version:	5.3		
Has patch:	No		
Complexity:	medium		
Subject:	Streamline Resource object API		

Description

The API and mechanism for resources is, for some time now, not as intended. What we basically need to achieve is:

- make sure that identical resources (same sha1) share the same file
- allow to create multiple combinations of sha1 / file name so that the original file name is preserved even if identical resources are uploaded
- still make sure that Resource objects with the same combination of sha1 / file name are unique (@identity for $hash and $filename)
- make sure that if a Resource object is deleted, the related file is deleted as well
- make sure that a Resource object is removed if it is not used anymore (no relation from another entity exists anymore)

For this to achieve we need to do (among other things) the following:

- remove the ResourcePointer class and move the getHash() and __toString() method to Resource
- implement some onDelete event handler for Resource objects which cleans up non-used files
- fix orphaned entity handling for aggregate members

The last point is especially important as it is an important feature for the overall persistence.

Subtasks:

Feature # 33587: Automatically remove unused Resources	**New**
Feature # 33469: Support for temporary Resources	**Rejected**

Related issues:

related to TYPO3.Flow - Bug # 36717: Remove Identity from Resource	**Resolved**	**2012-04-30**
related to TYPO3 Flow Base Distribution - Story # 42407: Asset Management	**New**	**2012-10-26**
related to Base Distribution - Work Package # 45003: Media Browser	**Accepted**	
blocked by TYPO3.Flow - Bug # 36804: Orphaned entities within aggregates are ...	**New**	**2012-05-03**

History

#1 - 2013-02-06 16:20 - Adrian Föder

- Estimated time set to 0.00

above, there are particularly two bullet points I want to comment on:

- make sure that a Resource object is removed if it is not used anymore (no relation from another entity exists anymore)
- fix orphaned entity handling for aggregate members

(both the last in each list)

Make sure to not consider resources always bound or related to entities. Especially in light of #33469, "Support for temporary Resources", Resources might also be, for example, temporary generated and minified CSS and JS which even must be published.

So in these cases the surveillance of orphans is even more complicated because lateron it can't be reliably said whether such a resource is still necessary or not.

While thinking about all of this, I saw a lot of "overlap" with the Caching framework. The Caching framework supports tagging, deletion, lifetime, ...

So maybe the Resource framework can joint venture the Caching framework in these regards. I could imagine some kind of a "PublishableFrontend(Interface)" or such, where a Resource is wired to a Cache entry.
Maybe publishing could be a method/feature of the (Cache-)FrontendInterface at all...

//addition: considering the (Cache-)FileBackend, it could carry an additional interface ResourcePublishable, which will result into a symlink created pointing to the cache entry.
Or, there won't be only a Resource object, but a network of a FileResource (being just like the current Resource), and a CacheResource. The fact that a Resource **always** intrinsically means a present file on disk is probably also unlucky.

Currently the literal identifier of a resource is its sha1 hash. I wanted to create a resource out of a curl-stream (incoming), which was not possible because I didn't know the final hash of course. How could this be achieved?

#2 - 2013-02-13 13:41 - Karsten Dambekalns

- Status changed from New to Accepted

- Assigned To set to Robert Lemke

#3 - 2013-08-02 17:03 - Adrian Föder

- Estimated time deleted (0.00)