Concept - Managing asset browser caching

Purpose

Note that this issue is basically identical to how assets such as javascript and css files are cached on the browser. So the solution should be the same or similar, and can be informed by how its handled for these types of resources.

The Problem

For many reasons assets should be cached by the client where possible:

- Less demand on server
- Less bandwidth usage on server
- Faster page load for client
- Improved page ranking by search engines SEO

"Normal" client caching technique is that you can set a header on a resource (Expires or Cache-Control) that specifies how long the browser should cache that resource, ie 1 day. The problem is that we don't know in advance when the resource is no longer valid, so these headers are of limited use. We really want the client to only grab the resources once, and then continue to use it until the resource changes. So practically, we'd like to set the Cache-Control to a year in the future.

Even if the browser implements a "Conditional Get" to check if the server resource date is newer then its cached version, this still has the expense of numerous http roundtrips.

When an asset is updated, then the UI or website must reflect this change as soon as possible, for these reasons:

- In the admin tool - a user must instantly see any updates that they make to an asset, otherwise the system appears broken & the user thinks that their change was not successful.
- In the page editor - a user wants to see the result of any changes that they made right away.
- On the live site - users should see the latest content that is available (where possible, balancing resource usage vs. content freshness)

Currently Magnolia public sites suffer in performance because they do not take advantage of far-future caching on images. See this page speed report as an example:

https://developers.google.com/speed/pagespeed/insights#url=http_3A_2F_2Fdemopublic.magnolia-cms.com_2Fdemo-project.html&mobile=false

Proposal

Design

- Set cache headers for far future caching so that clients and proxies maintain copies of asset.
- Add a "fingerprint" to the url of the asset. When we want the clients to get a new version of the asset, we change the fingerprint so that the url of the asset changes, and the cache treats it as a new file.

What to use as Fingerprint?

Fingerprint could be based on modification date of the file, or on a hash of the content, like MD5.

- We already use modification date for javascript and css.
- If we use modification date, the ideal would be the modification date of the media of the asset, not the modification date of any of the metadata like its title.
- If we use modification date, we must ensure that the modification date is the same across a public instance cluster - otherwise you might get different versions of the file depending on which server you hit.
- A content hash would ensure that the fingerprint ONLY changes when the content media actually changes.
- A content hash would be more resource intensive to compute then simply grabbing the date.

For now, we'll use modification date.

How is the fingerprint stored in the url?

Fingerprint could be in the actual file name, or it could be in a querystring parameter.

- At first glance, adding to querystring seems easiest.
- Resources on internet recommend changing filename instead of querystring, the main argument being that some proxies do not cache based on query string. (Squid < v2.7) This is an older version of squid, my opinion is that it would probably be ok to use query string.
- But it turns out to be very easy to alter the filename because magnolia does not actually grab the resource based on it's filename, but on the path!

So, we'll insert the fingerprint in the filename.

Relevant web resources:

- https://developers.google.com/speed/docs/best-practices/caching
  Recommendations: "Dont include a querystring in the URL for static resources" Squid up through version 3 do not cache resources with a ? in the url. (by default configuration)
- https://developers.google.com/speed/articles/caching
- http://www.stevesouders.com/blog/2008/08/23/revving-filenames-dont-use-querystring/
- http://guides.rubyonrails.org/asset_pipeline.html

Implementation

Implement in the assets app:

- Magnolia UI: DefaultImageProvider.getGeneratorImagePath() must add fingerprint to the imagePath it returns.

Implement in the page editor and live website:

- STK: InternalAssetVariation must add fingerprint to the link it sets.

Questions

- FarFuture Headers:
  - In which code should the caching header be set to the far future?
    - in the ImagingServlet.doGet ?
    - In the cache module? Where in the cache module?
      - See CacheHeadersFilter.java class. (See comments - appears to be supported already?)
        - But does this configuration only apply to public site? What about author site? What about assets displayed in the assets application? Ideally we could cache on the browser in this case as well.
  - Does far future caching need to be configurable?
    - Is there a case when one would not want far future caching?
- Fingerprinting code:
  - How best to not reproduce the fingerprinting code?
    - Currently in:
      - templatingkit/resources/Resource.java getFarFutureCachingTimeStamp()
      - magnolia main, LinkUtil.addFingerprintToLink()
    - I think the getFarFutureCachingTimeStamp() should use the FINGERPRINT_FORMAT defined in LinkUtil.
- What about images that are served from the resource directory? Is there a way to change their names with a fingerprint in the same way?