BeeCon 2016

Gotchas encountered while implementing an asynchronous transformation service for Alfresco

Lutz Horn <lutz.horn@ecm4u.de>
ecm4u GmbH http://www.ecm4u.de/
https://github.com/ecm4u

What I will talk about

- The Scenario
- Our Motivation
- What we've tried
- Gotcha: Transforming what exactly?
- Gotcha: We want to control if we transform!
- Gotcha: Content must not be lost!

The Scenario: Size

- Big Alfresco Community installation
- 200+ concurrent users
- 10 Million+ documents

The Scenario: Users

- Windows Users
- Access over CIFS and SharePoint
- MS Office is the client of choice
- Documents are MS Office documents

Our Motivation

- Render complex MS Office documents in an appealing way
- Alfresco becomes unresponsive when many documents are transformed at the same time. Fix this!
- Control the load, order, and results of concurrent transformations

What we've tried

- Implement job queue
- Policy on content update creates job
- Save dummy renditions
- Asynchronous worker on job queue
- Upload result to change renditions
- Cache extracted text for Solr

Transforming what exactly?

1st method in `ContentTransformer`

```
/**
 * @see #transform(ContentReader, ContentWriter, TransformationOptions)
 */
public void transform(ContentReader reader, ContentWriter writer) throws ContentIOException;
```

BeeCon 2016

Transforming what exactly?

2nd method in `ContentTransformer`

```
/**
 * Transforms the content provided by the reader and source mimetype
 * to the writer and target mimetype with the provided transformation options.
 * <p>
 * The transformation viability can be determined by an up front call
 * to {@link #isTransformable(String, String, TransformationOptions)}.
 * <p>
 * The source and target mimetypes <b>must</b> be available on the
 * {@link org.alfresco.service.cmr.repository.ContentAccessor#getMimetype()} methods of
 * both the reader and the writer.
 * <p>
 * Both reader and writer will be closed after the transformation completes.
 * <p>
 * The provided options can be null.
 *
 * @param  reader               the source of the content
 * @param  contentWriter        the destination of the transformed content
 * @param  options              transformation options, these can be null
 * @throws ContentIOException   if an IO exception occurs
 */
public void transform(ContentReader reader, ContentWriter contentWriter, TransformationOptions options)
    throws ContentIOException;
```

BeeCon 2016

Transforming what exactly?

Members of `TransformationOptions`

```
/** The source node reference */
private NodeRef sourceNodeRef;

/** The source content property */
private QName sourceContentProperty;

/** The target node reference */
private NodeRef targetNodeRef;

/** The target content property */
private QName targetContentProperty;
```

Transforming what exactly?

Patch `NodeContentGet`

```
long start = System.currentTimeMillis();
transformer.transform(reader, writer);
long transformDuration = System.currentTimeMillis() - start;
res.setHeader(TRANSFORM_DURATION_HEADER, String.valueOf(transformDuration));
```

```
// get the transformer
TransformationOptions options = new TransformationOptions();
options.setUse("index");
options.setSourceNodeRef(nodeRef);
```

```
long start = System.currentTimeMillis();
transformer.transform(reader, writer, options);
long transformDuration = System.currentTimeMillis() - start;
res.setHeader(TRANSFORM_DURATION_HEADER, String.valueOf(transformDuration));
```

Wish List 1

- Pass as much information as possible, you don't know who will want to use it.
- Provide a Push API to pass text to Solr.
- [more]

We want to control if we transform!

TransformerSelector

```
/**
 * Selects a transformer from a supplied list of transformers that appear
 * able to handle a given transformation.
 *
 * @author Alan Davis
 */
@AlfrescoPublicApi
public interface TransformerSelector
{
    /**
     * Returns a sorted list of transformers that identifies the order in which transformers
     * should be tried.
     * @param sourceMimetype
     * @param sourceSize
     * @param targetMimetype
     * @param options transformation options
     * @return a sorted list of transformers, with the best one first.
     */
    List<ContentTransformer> selectTransformers(String sourceMimetype, long sourceSize,
            String targetMimetype, TransformationOptions options);
}
```

We want to control if we transform!

`TransformerSelectorImpl`

Default transformer selector implementation, which sorts by priority and then by average transform time.

BeeCon 2016

We want to control if we transform!

- Patch Spring bean `contentTransformerRegistry`
- Implement own `TransformerSelector`
- Sort own `ContentTransformer`s on top of list of available `ContentTransformer`s

We want to control if we transform!

Now we can always be on top of the list of Transfomers based on our own logic and properties of the node.

Wish List 2

- Make it possible to control selection of Transfomers based on node properties
- Make Transformers configurable using a configuration logic even I can understand :)

Content must not be lost!

- How we wanted to control writing of renditions:
  - A **Policy** that listens on content updates,
  - writes dummy renditions,
  - and creates a job.
  - A **WS** that accepts a rendition and writes it.

Content must not be lost!

- When is content *really* written?
- Which events to ignore?
- Does the node still exist?
- MS Office uses a Shuffle mode!
- Two files are swapped!

Content must not be lost!

Beware of `RenditionedAspect`!

```
/**
 * Renditioned aspect behaviour bean.
 * When any node with the renditioned aspect has a property updated, then all
 * associated renditions are eligible for re-rendering.
 * Each rendition (as identified by the name in its rn:rendition association) will
 * be loaded and if the renditionDefinition exists, the rendition will be updated
 * asynchronously, subject to the defined update policy.
 *
 * @author Neil McErlean
 * @author Roy Wetherall
 */
public class RenditionedAspect implements NodeServicePolicies.OnUpdatePropertiesPolicy,
                                           CopyServicePolicies.OnCopyNodePolicy
```

```
Action deleteRendition = actionService.createAction(DeleteRenditionActionExecuter.NAME);
deleteRendition.setParameterValue(DeleteRenditionActionExecuter.PARAM_RENDITION_DEFINITION_NAME, rendDefn.getRenditionName());
rendDefn.setCompensatingAction(deleteRendition);

renditionService.render(sourceNodeRef, rendDefn, new RenderCallback()
```

BeeCon 2016

Content must not be lost!

- So we have *two* policies that both trigger on the same events and write renditions.
- MS Office Shuffle mode uses *two* nodes for one logical document.
- Failure in one of the policies can lead to a `bin` file never be written.
- => Real loss of content!

BeeCon 2016

Content must not be lost!

- Extend RenditionedAspect
- Only call super implementation if we don't care about the node or if there is no current job for it.
- Thus avoid two competing policies working on renditions of the same node.

Lessons learned

- Somtimes (often?) it is necessary to patch Alfresco if you want to be able to write extension code.
- Alfresco may decide not to run your code because of obscure reasons.
- Always be prepared that existing Alfresco code is competing with your code.

Thanks!

BeeCon 2016