WritingForCacheEfficiency

Pipeline components can be made cacheable by implementing `CacheableProcessingComponent`. This allows the component to give a cache key and validity used to avoid useless reprocessing if the pipeline result would be unchanged.

There are a number of existing Cocoon components that implement this interface and that you can take as samples when writing your own.

However, here are a few tips that can help to achieve maximum efficiency.

Avoiding uncessary operations

Achieving maximum performance to deliver cached content needs some careful coding of cacheable components.

Looking at `TraxTransformer.setup()` (as of 13 may 2003), we can see that it gets the validity *and* and a TransformerHandler at the same time. And if the validity is still valid, the TransformerHandler is simply not used since the content is retrieved from the cache. So I hacked a bit to separate these, and obtained again a speed increase ranging from 5% to 30%.

This leads to some importants recommendations in order to achieve the maximum cache efficiency : **the `setup()` method must avoid performing operations that are necessary only if the content is not cached**. Otherwise, its just a waste of speed to deliver cached content.

Here's a reminder of the various steps that occur when handling a request :

1. the sitemap is executed, meaning we create a pipeline object, and pipeline components : generator, transformers, and serializer.
2. the `setup()` method of all pipeline components is called (including serializers that implement SitempaModelComponent))
3. the serializer is asked for the response mime-type
4. the `getKey()` method of all pipeline components is called

Knowing the key, the pipeline can get the associated cache entry and its validity. If the cache validity either is invalid or needs a fresh validity object to be compared with, then :

1. the `getValidity()` method of all pipeline components is called

The pipeline can then know if the cache entry is valid. If it's valid, it delivers the cached content. If it's invalid, then :

1. the pipeline is connected. This means `setXMLConsumer()` is called on transformers and `setOutputStream()` is called on the serializer
2. the generator's `generate()` method is called, and starts the SAX stream processing, resulting first in `startDocument()` being called on transformers and serializer.

What we can see above, is that we must defer as much as possible to points 5 or 6 the creation or lookup of resources that are used to process the content. Doing it before is only waste of resources. And that's what TraxTransformer is doing : it creates a TransformerHandler at point 2.

Be aware of that and inspect cacheable components for possible enhancements. This is key for an increased performance !

Cache validity depends on pipeline execution

Some pipeline components have a validity that depends on the execution of the pipeline. Some examples include :

- the DirectoryGenerator : validity is the list of files whose XML representation is produced,
- the TraxTransformer : we need to track source included with the `document()` function.

How can such components be made cacheable, since `getValidity()` is called before pipeline execution (see above) ?

The solution used in the DirectoryGenerator is for `getValidity()` to return a validity object that is initially just an empty placeholder, and is filled with file names and modification dates during the pipeline execution phase. When pipeline execution is finished, the produced content goes into the cache with a validity object which is correctly filled.

Et voilà ! Cache validity is actually computed during pipeline execution !

– SylvainWallez