SolrAndHTTPCaches

Let Solr play nicely with HTTP caches

⚠️ Solr1.3

Configuration

Solr

Solr honors following request header elements:

- If-None-Match
- If-Match
- If-Modified-Since
- If-Unmodified-Since

Solr emits following response header elements:

- Last-Modified
- ETag
- Expires
- Cache-Control

⚠️ Solr only emits cache header elements for GET and HEAD requests. The HTTP standard does not allow cache related headers for POST requests. POST requests are not cached by standard compliant shared caches!

Exactly how Solr behaves can be configured in the "httpCaching" section of solrconfig.xml.

Rules of thumb for the *Cache-Control* header:

- When you need to make sure that your application gets the latest version of a Solr response:
  - Add *must-revalidate* to the Cache-Control header setting. This forces the shared cache to check on every request that the Solr index has not changed.
- When you don't want shared caches to cache Solr's response:
  - Add *no-cache, no-store* to the Cache-Control header setting. This forces the shared cache to always get a fresh response from Solr.
- When you update your index only from time to time (once an hour, once a day, once a week, ...):
  - Add *max-age=<half the update interval in seconds>* to the Cache-Control header setting. For example if you update every minute you should set max-age to 30 seconds.
- When you only want browser caches (the spec. calls this "private caches") to cache the Solr response:
  - Add *private* to the Cache-Control header setting.

When we talk about *shared caches* this also means browser caches.

All these parameters can be combined, of course (making more or less sense, of course). A good way to start is, for example, *max-age=600, must-revalidate*. If all intermediate caches work spec. compliant then the behaviour of your application does not change. Mixing *max-age* with *no-cache, no-store* makes no sense, of course. The paranoid can choose *private, max-age=0, no-cache, no-store*.

Particularly big corporations use ancient proxy software (some still use Netscape Proxy Server), so you might run into trouble when you enable this feature. It is not very likely but you never know. Solr does everything to avoid such problems because it emits *HTTP 1.0* and *HTTP 1.1* compliant HTTP headers.

Popular Cache Implementations

Squid Cache

No special configuration is needed for Squid Cache. The only thing you need to check is that the configuration parameter cache does not contain the ? pattern (this prevents all GET requests with parameters from beeing cached). The *cache* parameter tells Squid what never gets cached.

Apache HTTP server

Solr plays nicely with the caching module of the Apache web server as well. Read the Caching Guide to get it working.

Further readings

- Caching in HTTP
- How to Optimize Your Site With HTTP Caching