CoreAdmin

Quick Review: What are Multiple Cores?

Multiple cores let you have a single Solr instance with separate configurations and indexes, with their own config and schema for very different applications, but still have the convenience of unified administration. Individual indexes are still fairly isolated, but you can manage them as a single application, create new indexes on the fly by spinning up new SolrCores, and even make one SolrCore replace another SolrCore without ever restarting your Servlet Container. See MultipleIndexes

Core Administration

⚠️ Solr1.3

Since Solr1.3, SolrCore can optionally be managed at runtime. Additionally, Solr allows multiple SolrCore instances to run within a single web-app. The cores can be dynamically managed via the CoreAdminHandler. For alternative ways to manage multiple indices, see MultipleIndexes.

Configuration

As of ⚠️ Solr4.4 there will be an optional new structure for solr.xml that will be mandatory for 5.0. See Solr.xml 4.4 and beyond and Core Discovery (4.4 and beyond).

As of ⚠️ Solr5.0 this new structure will be mandatory and cores will be discovered by walking SOLR_HOME or coreRootDirectory - see links above.

CoreAdminHandler

The CoreAdminHandler is a special SolrRequestHandler that is used to manage existing cores. Unlike normal SolrRequestHandlers, the CoreAdminHandler is not attached to a core, it is configured in solr.xml. A single CoreAdminHandler exists for each web-app

To enable dynamic core configuration, make sure the *adminPath* attribute is set in solr.xml. If this attribute is absent, the CoreAdminHandler will not be available.

STATUS

Get the status for a given core or all cores if no core is specified:

http://localhost:8983/solr/admin/cores?action=STATUS&core=core0

http://localhost:8983/solr/admin/cores?action=STATUS

CREATE

Creates a new core based on preexisting instanceDir/solrconfig.xml/schema.xml, and registers it. If persistence is enabled (persist=true), the configuration for this new core will be saved in 'solr.xml'.

http://localhost:8983/solr/admin/cores?action=CREATE&name=coreX&instanceDir=path_to_instance_directory&config=config_file_name.xml&schema=schema_file_name.xml&dataDir=data

instanceDir is a required parameter. config, schema & dataDir parameters are optional. (Default is to look for solrconfig.xml/schema.xml inside instanceDir. Default place to look for dataDir depends on solrconfig.xml.)

⚠️ Solr3.4 Core properties can be specified when creating a new core using optional `property.name=value` request parameters, similar to `<property>` tag inside solr.xml.

⚠️ Solr4.3 Optional parameters:

- loadOnStartup=[**true**|false] - whether to load the core when Solr starts or wait until the first time it's referenced.
- transient=[true|**false**] - whether the core can be automatically unloaded if the number of transient cores exceeds the transientCacheSize parameter that may be specified in the <cores> tag. See [Solr.xml]

The behaviour of the CREATE action when passed the name of a pre-existing core depends on the Solr version:

- Prior to Solr 4, a new core is created in the background. While it is initializing, the old core will continue to accept requests. Once it has finished, all new request will go to the "new" core, and the "old" core will be unloaded.
- In Solr 4.0 to 4.2, the above behaviour still holds, but is buggy, and clients should use the RELOAD action instead.
- In Solr 4.3 and above, an error is returned, and RELOAD must be used.

RELOAD

Load a new core from the same configuration as an existing registered core. While the "new" core is initalizing, the "old" one will continue to accept requests. Once it has finished, all new request will go to the "new" core, and the "old" core will be unloaded.

http://localhost:8983/solr/admin/cores?action=RELOAD&core=core0

This can be useful when (backwards compatible) changes have been made to your solrconfig.xml or schema.xml files (e.g. new `<field>` declarations, changed default params for a `<requestHandler>`, etc...) and you want to start using them without stopping and restarting your whole Servlet Container.

Important Note About Some Configuration Changes

Starting with Solr4.0, the RELOAD command is implemented in a way that results a "live" reloads of the SolrCore, reusing the existing various objects such as the SolrIndexWriter. As a result, some configuration options can not be changed and made active with a simple RELOAD...

- `IndexWriter` related settings in `<indexConfig>`
- `<dataDir>` location

See SOLR-3592 for more background.

RENAME

Change the names used to access a core. The example below changes the name of the core from "core0" to "core5".

http://localhost:8983/solr/admin/cores?action=RENAME&core=core0&other=core5

SWAP

Atomically swaps the names used to access two existing cores. This can be useful for replacing a "live" core with an "ondeck" core, and keeping the old "live" core running in case you decide to roll-back.

http://localhost:8983/solr/admin/cores?action=SWAP&core=core1&other=core0

UNLOAD

Removes a core from Solr. Existing requests will continue to be processed, but no new requests can be sent to this core by the name. If a core is registered under more than one name, only that specific mapping is removed.

http://localhost:8983/solr/admin/cores?action=UNLOAD&core=core0

⚠️ Solr3.3 An optional boolean parameter "deleteIndex" can be used to delete the index on core unload.

http://localhost:8983/solr/admin/cores?action=UNLOAD&core=core0&deleteIndex=true

⚠️ Solr4.0 Two more optional parameters are "deleteDataDir" and "deleteInstanceDir" on core unload.

- deleteDataDir removes "data" and all sub-directories
- deleteInstanceDir removes *everything* related to the core, the index directory, the configuration files, etc. This command would remove the directory core0 and all sub-directories. NOTE: there is a bug in 4.0 SOLR-3984 currently (4.0.0) that prevents this from working unless you specify the absolute path in your <core.../>.

```
http://localhost:8983/solr/admin/cores?action=UNLOAD&core=core0&deleteIndex=true
```

These three "delete*" commands form a hierarchy. "deleteInstanceDir" will do what both "deleteDataDir" and "deleteIndex" do and much more, so use cautiously. "deleteDataDir" will also "deleteIndexDir". You should only need to specify one.

LOAD

⚠ **not implemented yet! Use CREATE**

So far, no use cases have been presented for a LOAD command that aren't satisfied by using CREATE so it's doubtful that a separate LOAD command will be implemented unless such a use-case is found.

This will load a new core from an existing configuration (will be implemented when cores can be described with a lazy-load flag).

?persist=true will save the changes to solr.xml

http://localhost:8983/solr/admin/cores?action=LOAD&core=core0

MERGEINDEXES

⚠ Solr1.4

Merge indexes into a another index. This is described more fully at MergingSolrIndexes.

SPLIT

⚠ Solr4.3

Splits an index into two or more indexes. It accepts the following parameters:

- "core" - The core whose index is to be split
- "path" - The file path to which the pieces of the "core"'s index will be written (multi-valued parameter)
- "targetCore" - The target solr core (which must already exist) to which the pieces of the split index will be merged (multi-valued parameter)

Either "path" or "targetCore" must be specified (but not both). At least two values for "path" or "targetCore" must be specified.

Example:

- http://localhost:8983/solr/admin/cores?action=SPLIT&core=core0&targetCore=core1&targetCore=core2
- http://localhost:8983/solr/admin/cores?action=SPLIT&core=core0&path=/path/to/index/1&path=/path/to/index/2

This command is used as part of the SPLITSHARD SolrCloud Collection API but it can be used for non-cloud Solr cores as well. When used against a non-cloud core, this action will split the source index into parts containing an equal number of documents.

Known Issues

Lucene's BooleanQuery maxClauseCount is a static variable, making it a single value across the entire JVM. Whichever Solr core initializes last will win the setting of the solrconfig.xml's maxBooleanClauses value. Workaround, set maxBooleanClauses to the greatest value desired in *all* cores.