TaskList

This is a list of ideas for improving Solr.

All users should feel free to add new ideas to this page, or add links to other wiki pages containing more involved designs.

Users should also feel free to open "New Feature", "Improvement", or "Wish" issues in Jira – particularly if they already have code that makes progress towards the idea.

Many of the ideas on this page have been discussed on the Solr mailing lists, you should search there for more information.

Simple Non-Invasive Tasks

This section is for ideas that are relatively straight forward or don't involve major changes to the Solr codebase. People who are eager to "give back" to the Solr community but don't have a lot of familiarity with the Solr code base may be interested in taking on these Tasks...

- the bf param of dismax should be smarter then just splitting on whitespace (ie: don't split in the middle of parens)
- "Build Instructions" should probably be moved out of README.txt and into a BUILD.txt and expanded on.
- Create a Powered_By_Solr icon that people can include in their applications if they so choose.
- all source code in "tgz" based releases should be processed by ant's `<fixcrlf>` in case someone prepares a release on a windows box
- alternate ways of indexing?
  - over JMS
- a DateTime field (or Query Parser extension) that allows flexible input for easier human entered queries
- allow alternate format for date output to ease client creation of date objects?
- support for max disjunction and minNrShouldMatch in query parser (really a Lucene item)
- UnitsFilter... 17" => 17 inch, etc
- Admin query interface: add highlighting options, query writer options, facet options (see also SOLR-67)
- Documentation
  - Add to the existing tutorial, or write new tutorials discussing out of the box features...
    - highlighting
    - dismax handler
    - the various output formats
    - simple facet counts
    - DataImportHandler
    - CSV Loading
  - result XML format - needed, or self-explanatory?
  - Java Docs
    - good overview.html
    - package.html for every package
    - class level documentation for every class
    - detailed method javadocs for every method in all of the "pluggable" classes and every method in a key class used when writing a request handler...
      - SolrCache ... in progress
      - SolrEventListener
      - UpdateHandler
      - FieldType ... in progress
      - SchemaField
- SolrQueryParser Configuration in schema.xml
  - make more options configurable via schema.xml besides operator (discussion)
  - refactor option setting into a utility (possibly in IndexSchema) so people constructing a SolrQueryParser instance directly get the built in defaults. (discussion)
- Live demo server or application (perhaps host on apache lucene zone)
  - Mailing List Index? (No urgent need, since Lucid has a great such index already)
- Apache Whirr is automated deployment for app-level services like Solr. As you can see ElasticSearch is supported but Solr is not: Services and Cloud Providers. Go Solr! Beat ElasticSearch!
- Caches have fallen behind changes in index-handling. When a commit changes a few segments, it should be possible to throw away only the document cache entries for the removed segments.
- Another cool feature ElasticSearch has is 'percolation'- real-time alerts. After each commit it runs a search on only the new segments. If any new documents match the search, it does an HTTP call and posts the new document ids. This is profoundly useful, and Solr needs it.

Big Ideas for The Future

This section should be used for ideas that are more involved and may require major changes to the Solr codebase, and definitely should involve a lot of discussion among developers about the appropriate way to tackle them...

- Robust and configurable field aliasing and globbing support: FieldAliasesAndGlobsInParams
- A more powerful query language allowing one to express complicated logic without resorting to a custom Java query handler plugin.
- Make use of HiveMind or Spring for configuration & dependency injection
- Implement some ideas for ComplexFacetingBrainstorming or DynamicallyCalculatedFacetRanges
- Implement some ideas to MakeSolrMoreSelfService
- support for composite keys ... either with some explicit change to the `<uniqueKey>` declaration or perhaps just copyField with some hidden magic that concats the resulting terms into a single key Term
- investigate this "SynonymQuery" and how SynonymFilter might be modified to set token types to trigger it's use automatically in SolrQueryParser
- FederatedSearch
- refactor and separate update XML parsing from update handling... possibly implement support for JSON updates.

    - [DONEish?] the refactoring is done, but still no JSON update support

- [IN PROGRESS] refactor all of the JSP pages into request handler so a JDK/JSP compiler isn't needed (the current JSPs are very sparse on presentation, and use no custom tags, so there is almost no advantage to them being JSPs)

- Better handling of arbitrary XML charsets: see SOLR-96
- Better support for tagging: UserTagDesign
- RealtimeSearch and updates
- HadoopRmi based distributed search
- Automatic redundant failover and coordincation of SOLR servers using a distributed naming service (such as Zookeeper)
- New DocumentProcessing framework (aka pipeline/chain) which is robust, scalable (multi node) and easily extensible
- VariableRangeGaps discusses SOLR-2366 and a more flexible range facet
- Support multiple files for each of the core configuration files: solrconfig.xml, schema.xml and solr.xml. Find and load all of these files dynamically, and allow all to contribute configuration information. This is a much more flexible alternative to using XInclude in solrconfig.xml etc. Also, it allows Solr to stop rewriting solr.xml, which is a very bad design.