MakeSolrMoreSelfService

This is a collection of ideas about ways to make individual installations of Solr more "Self Service" for clients.

(A lot of these ideas come out of an informal discussion that happened at CNET a little while before Solr was open sourced. We were talking about things that could be done so that all you needed to "discover" an installation of Solr - and what could be done with it - was a link to the /admin screen. From there you could get all the info you needed without wondering where it might be documented externally.)

Misc

Some people feel that there should always be a guaranteed method for using the StandardRequestHandler – either by not specifiying a `qt` (even if some other handler is configured with the name "standard") or by using qt=standard (and attempting to register any handler by that name is an error)

Configuration

General Index Documentation

The `<admin>` block of solrconfig.xml should allow for more high level documentation about what the index is, what it contains, who maintains it, and how it's maintained. This info should be displayed on the `/admin` screen. Something like this perhaps...

```
<admin>
  <maintainer email="foo@bar.com">Chris Hostetter</maintainer>
  <!-- at least a minimal subset of html should be supported in
       indexDescription -->
  <indexDescription>
    <p>
    This index contains product data from the
    <b>personal electronics database</b>.
    It is built using the PEDBuilder, and updated hourly by the PEDUpdater.
    </p>
    <p>
    To see the most recently published products, use the
    <a href="../select/?qt=pedfoo&amp;recent=30">pedfoo handler</a>
    </p>
  </indexDescription>
  <link href="http://docserver.bar.com/ped/solr-index">index design</doc>
  <link href="http://svn.bar.com/ped/builder/">builder source code</doc>
  <link href="http://svn.bar.com/ped/updater/">updater source code</doc>
</admin>
```

Descriptions

Just about everything in solrconfig.xml and schema.xml file should support a `description="..."` free form text attribute to allow more exact (and machine readable) documenting of why things are the way they are (without relying solely on XML comments).

Specific things that would really be handy to have this...

- schema.xml
  - fieldtypes, analyzeries, tokenfactories, filterfactories.
  - fields, dynamicFields, copyFields
  - uniqueKey, defaultSearchField
  - similarity
- solrconfig.xml
  - caches, caching options
  - updateHandler, listeners (postCommit)
  - query listeners (newSearcher, firstSearcher)
  - request handlers
  - defaultQuery, pingQuery, healthcheck

Much of this info would make sense to be surfaced on the `/admin/registry.jsp` JSP, some of it will come in handy in the suggestions below...

Request Handler Param Docs

In addition to the `<init>` options that Query Handlers can use anyway they want, there should be a mechanism when registering handlers to specify what query params it supports, with descriptions, and some basic info on how they should be displayed in a form.

Perhaps something like...

```
<requestHandler name="example" class="myorg.mypkg.MyRequestHandler"
 description="This is my handler, it is not yours"
>
  <params>
    <param name="q"
           description="main query, in lucene parser syntax"
           type="text"
           samplevalue="+content:rad +author:me" />
    <param name="sort"
           description="sort options"
           type="text"
           samplevalue="score desc name asc" />
    <param name="rows"
           description="how many rows you want back"
           type="int"
           samplevalue="10" />
    <param name="behavior"
           description="what kind of behavior do you want?"
           type="list"
           samplevalue="a" >
       <li val="a">Type A Behavior</li>
       <li val="b">Some other type of behavior</li>
       <li val="how now brown cow" /><!-- use val as label -->
    </param>
  </params>
  <!-- the rest of these options are init params for the plugin -->
  <int name="myparam">1000</int>
  <float name="ratio">1.4142135</float>
  <arr name="myarr"><int>1</int><int>2</int></arr>
  <str>foo</str>
</requestHandler>
```

At first glance, it may seem like this info should be returned by some method in the SolrRequestHandler interface, but i think it makes more sense if the person registering the handler gets to specify what options are "publicly" advertised for the specific instance of the handler.

Advanced Search Form

`/admin/form.jsp` currently has a hard coded list of params, regardless of which plugin is used.

Assuming the param Configuration information described above is added to the solrconfig.xml, then the behavior of form.jsp could be driven by the `<params>` specified for the default handler, and an optional `qt` param could change the params displayed based on the handler selected (e.g. `form.jsp?qt=foo`). In this case, displaying the description of the handler would also be useful.

In addition, form.jsp should look at it's query params for any options that match the params of the specified `qt` and change the default form values accordingly (this would allow people to link to the form with values that override the defaults)

The main `/admin` screen should also be changed, so instead of (or in addition to) the "Full Interface" link to `form.jsp`, there is a form with a pulldown listing each handler `qt` option.

The bottom of `form.jsp` may also be a good place to list all of the registered handlers, with their descriptions, and the info from the `SolrMBean` interface methods (or maybe this should be a separate page). Each should have a link back to `form.jsp?qt=their_qt`

Schema Explorer

Having a link to the `schema.xml` file from the `/admin` screen is useful, but given the way fields can inherit/override options form their fieldtype, it's not always easy to understand what you are looking at.

A Schema Explorer page should exist, with features like...

- list all field types "with details"
- list of field types "with details" by major option...
  - indexed
  - stored
  - termVectors
  - multiValued
  - omitNorms

- list of all field type backing classes (i.e. SortableIntField, DateField, TextField, etc...) found in a fieldtype. For each class provide a list of all fieldtypes "with details" using that class.
- list of all fields "with details"
- list of all dynamic fields "with details"

Whenever a field type is displayed "with details", show...

- the description
- backing class
- options (i.e.: `omitNorms`, `stored`, `positionIncrimentGap`, etc...)
- analyzer and/or tokenizer/filter chain if they exist
    - descriptions of each if they exist
- A list of the fields that using this fieldtype (with some indication whether they override any options) and a link to their details.

Whenever a field (or dynamic field) is displayed "with details", show...

- the description
- the fieldtype
- the backing class
- options (whether explicit, or inherited from fieldtype)
- analyzer and/or tokenizer/filter chain if they exist
    - descriptions of each if they exist
- list of any fields that this field copies from
- list of any fields that this field copies to
- link to `analysis.jsp` for this field (this should work even with suffix/prefix dynamic fields)

Similarity Info

similarity.jsp

Along the same links as `analysis.jsp`, it would be useful if there was a simple URL that helped understand what the registered `Similarity` class was doing. Each of the methods could be represented by a small form for entering inputs, and the results of the function calls would be returned.

In the case of `lengthNorm`, both the raw value returned, as well as the value after it has been passed through `decodeNorm(encodeNorm(lengthNorm(...)))` should be returned.

For functions ("f") that take in integer or float arguments, the form should allow a min/max/increment triples to be specified, and should return the list resulting from...

```
for (int i = min; i <= max; i+=increment) {
   list.add( f(i) );
}
```

...so that it's easy to see what the functions do across a range of values.

If we really wanted to go all out, we could pick a graphing library to include as an optional jar, and if it's installed display graphs of the values between min /max

analysis.jsp changes

When displaying the tokens resulting from "Indexing" analysis, the number of tokens (and the field name) could be passed to `decodeNorm(encodeNorm(lengthNorm(...)))` to display what the lengthNorm would be for documents that had that text as it's whole value. (this should check `omitNorms` for the specified field of course).

When displaying the the tokens resulting from "Query" analysis, the idf for each Term (and the idf form all of the terms as a phrase) can be displayed.