Faceted Searching With Apache Solr

October 13, 2006
Chris Hostetter
hossman – apache – org
http://incubator.apache.org/solr/
What is Faceted Searching?
Example: Epicurious.com
Example: Nabble.com

Lucene...
...java-user-unsubscribe@... For additional commands, e-mail: java-user-help@......
in Lucene - Java Users on Jun 21 by Bruce-34 - replies: 1

Lucene
...created. Since I need to be continuously adding files indice, thus not if Lucene does what I need. My language is the Spanish does...
in Lucene - Java Users on May 17 by Alberto Marguillfes - replies: 1

Lucene
Hi again I want to use Lucene with a french website. If I search alézia, Lucene find my data, but if I search alesia, I have no answer. Do...
in Java - Dev on Jul 13, 2005 by Nicolas Lafaury - replies: 3

Lucene
Hi list, can i use Lucene in OpenCms 6 to provide a Search in a password restricted area? I have some free content and some sites that are only...
in OpenCMS - Dev on Dec 21, 2005 by shule1212 - replies: 1

Lucene
...enterprise level applications) - would anyone be interested if I embarked on integrating Lucene into FarCry as an alternative to Verity? I am...
in FarCry - Dev on Jun 05, 2005 by Robertson-Ray, Neil (RX) - replies: 2

Lucene faster on JDK 1.5?...
...unsubscribe, e-mail: java-user-unsubscribe@... > > For additional commands, e-mail:
Example: CNET.com

Microsoft LifeCam VX-6000
The Microsoft LifeCam VX-6000 offers unique features such as the ability to post photos directly to a blog, but its video effects and image quality don't stand up to that of competing Webcams from veteran manufacturers.

Specs: Drivers & Utilities

Add to my products  New!  What is this?
Aka: “Faceted Browsing”

"Interaction style where users filter a set of items by progressively selecting from only valid values of a facet classification system"

- Keith Instone, SOASIS&T, July 8, 2004
Key Elements of Faceted Search

• No hierarchy of options is enforced
  – Users can apply facet constraints in any order
  – Users can remove facet constraints in any order

• No surprises
  – The user is only given facets and constraints that make sense in the context of the items they are looking at
  – The user always knows what to expect before they apply a constraint
Explaining My Terms

• Facet: A distinct feature or aspect of a set of objects; “a way in which a resource can be classified”

• Constraint: A viable method of limiting a set of objects
Dynamic Taxonomy? No.

- Bad Description
- Taxonomy implies a hierarchy of subsets
- Hierarchy implies ordered usage of constraints
Why Is Faceted Searching Hard?

- **LOTS** of set intersections
- All permutations can't be easily precomputed
What is Solr?
Elevator Pitch

"Solr is an open source enterprise search server based on the Lucene Java search library, with XML/HTTP APIs, caching, replication, and a web administration interface."
What Does That Mean?

• Information Retrieval application
• Java5 WebApp (WAR) with a web services-ish API
• Uses the Java Lucene search library
• Initially built at CNET
• Now an Apache Incubator project
Lucene Refresher

- Lucene is a full-text search library
  - Maintains inverted index: terms -> documents
- Add documents to an index via IndexWriter object
  - A document is a collection of fields
  - No config files, dynamic field typing
  - Text analysis performed by Analyzer objects
  - No notion of "updating" or "replacing" an existing document
- Search for documents via IndexSearcher object
  
  Hits = search(Query, Filter, Sort, topN)

- Scoring: \( tf \times idf \times lengthNorm \)
Solr in a Nutshell

- Index/Query via HTTP and XML
- Comprehensive HTML Administration Interfaces
- Scalability - Efficient Replication to Other Solr Search Servers
- Extensible Plugin Architecture
- Highly Configurable and User Extensible Caching
- Flexible and Adaptable with XML configuration
  - Data Schema with Dynamic Fields and Unique Keys
  - Analyzers Created at Runtime from Tokenizers and TokenFilters
Example: Adding a Document

HTTP POST /update

<add><doc>
  <field name="article">05991</field>
  <field name="title">Apache Solr</field>
  <field name="subject">An intro...</field>
  <field name="cat">search</field>
  <field name="cat">lucene</field>
  <field name="body">Solr is a full...</field>
  <field name="inStock">true</field>
</doc></add>
Example: Execute a Query

HTTP GET
/select/?qt=foo&wt=bar&start=0&rows=10&q=solr

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <responseHeader>
    <status>0</status><QTime>1</QTime>
  </responseHeader>
  <result numFound="1" start="0">
    <doc>
      <arr name="cat">
        <str>lucene</str><str>search</str>
      </arr>
      <bool name="inStock">true</bool>
      <str name="title">Apache Solr</str>
      <int name="popularity">10</int>
    </doc>
  </result>
</response>
Example: SimpleRequestHandler

```java
public void handleRequest(SolrQueryRequest req,
                         SolrQueryResponse rsp) {
    try {
        Query q = QueryParsing.parseQuery
            (req.getQueryString(), req.getSchema());

        DocList results =
            req.getSearcher().getDocList
            (q, (Query)null, (Sort)null,
             req.getStart(), req.getLimit());

        rsp.add("simple results", results);
        rsp.add("other data", new Integer(42));
    } catch (Exception e) {
        rsp.setException(e);
    }
}
```
DocLists and DocSets

- **DocList** - An ordered list of document ids with optional score
  - A subset of the complete list of documents actually matched by a Query
- **DocSet** - An unordered set of Lucene Document Ids
  - Typically the complete set of documents matched by a query
  - Multiple implementations optimized for different size sets
  - Foundation of Faceted Searching in Solr
Caching

- IndexSearcher's view of an index is fixed
  - Aggressive caching possible
  - Consistency for multi-query requests

Types of Caches:
- filterCache: Query => DocSet
- resultCache: (Query,Sort,Filter) => DocList
- documentCache: docId => Document
- userCaches: Object => Object
  - application specific, custom query handlers
Smart Cache Warming

Static Warming Requests

Live Requests

On-Deck Solr IndexSearcher

User Cache

Filter Cache

Result Cache

Doc Cache

Request Handler

Registered Solr IndexSearcher

User Cache

Filter Cache

Result Cache

Doc Cache

Regenerator

Regenerator

Regenerator

Autowarming

Autowarming – warm n MRU cache keys w/ new Searcher

Field Cache

Field Norms
Case Study

CNET's First Solr Powered Page
## Old Crappy Version

<table>
<thead>
<tr>
<th>Product</th>
<th>Editors' rating</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microsoft LifeCam VX-6000</strong></td>
<td><strong>7.0</strong></td>
<td>Email me when this product is available</td>
</tr>
<tr>
<td>The Microsoft LifeCam VX-6000 offers unique features such as the ability to post photos directly to a blog, but its video effects and image quality don't stand up to that of competing Webcams from veteran manufacturers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review date: 06/14/2006 Release date: 06/13/2006 Specs: Drivers &amp; Utilities ▶ CNET editor's take</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Creative Live Cam Voice</strong></td>
<td><strong>7.2</strong></td>
<td>Email me when this product is available</td>
</tr>
<tr>
<td>With beefed-up audio features, the Creative Live Cam Voice is one of the best overall cameras for IM and Internet-based voice conferencing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review date: 05/16/2006 Release date: 05/16/2006 Specs: Yahoo! Messenger - Gray -1,300,000 pixels ▶ CNET editor's take</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WILife LukWerks Starter Kit</strong></td>
<td><strong>7.1</strong></td>
<td>Email me when this product is available</td>
</tr>
<tr>
<td>The WILife LukWerks system is easy to configure and use, but the software can be cantankerous. Potential users may suffer sticker shock, but it's a deal compared to professionally installed security systems.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review date: 03/28/2006 Release date: 02/01/2006 Specs: Drivers &amp; Utilities ▶ CNET editor's take</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shiny New Faceted Version

Find by price
- $90 - $150 (18)
- $150 - $250 (20)
- $250 - $320 (15)
- $320 - $450 (15)
- $450 - $600 (17)
- See all prices

Find by manufacturer
- Axis Communications (42)
- Logitech Inc. (41)
- 4XEM Corporation (21)
- Panasonic (19)
- Creative Labs Inc. (18)
- See all manufacturers

Find by audio input type
- Microphone (94)
- None (92)
- Headset (7)

Or find by
- Compatibility
- Connector type
- Interface type

Microsoft LifeCam VX-6000
The Microsoft LifeCam VX-6000 offers unique features such as the ability to post photos directly to a blog, but its video effects and image quality don’t stand up to that of competing webcams from veteran manufacturers.

Specs: Drivers & Utilities

Add to my products New! What is this?

Creative Live Cam Voice
With beefed-up audio features, the Creative Live Cam Voice is one of the best overall cameras for IM and Internet-based voiceconferencing.

Specs: 1,300,000 pixels, Yahoo! Messenger, Skype
Category Metadata

- Category ID and Label
- Category Query
- Ordered List of Facets
  - Facet ID and Label
  - Facet "Display Type"
  - Ordered List of Constraints
    - Constraint ID and Label
    - Constraint Query
Key Features We Needed In Solr

- Loose Schema with Dynamic Fields
- Efficient implementation of sets and set intersection
- Aggressive set caching
- Plugin Architecture
RequestHandler Pseudo-Code

```java
Document catMetaDoc = searcher.getFirstMatch(categoryDocId)
Metadata m = parseAndCacheMetadata(catMetaDoc, searcher).clone()

DocListAndSet results = searcher.getDocListAndSet(m.catQuery, ...)

response.add(results.docList)

foreach (Facet f : m) {
    foreach (Constraint c : f) {
        c.setCount(searcher.numDocs(c.query, results.docSet))
    }
}

response.add(m.dumpToSimpleDatastructures())
```
Conceptual Picture

getDocListAndSet(Query, Query[], Sort, offset, n)

- computer_type: PC
- memory: [1GB TO *]
- computer
- price asc
- proc_manu: Intel
- proc_manu: AMD
- price: [0 TO 500]
- price: [500 TO 1000]
- manu: Dell
- manu: HP
- manu: Lenovo

numDocs() = 594
= 382
= 247
= 689
= 104
= 92
= 75
XML Response

- <response>
  - <responseHeader>
    <status>0</status>
    <QTime>17</QTime>
  </responseHeader>
  + <result name="products" numFound="5461" start="0"></result>
  - <lst name="metadata">
    - <lst name="100021">
      <int name="rankDir">1</int>
      <int name="fornelement">10</int>
      + <lst name="values">
        <int name="datatype">3</int>
        <int name="rating">94</int>
        <str name="name">Price</str>
        <int name="attributeld">100021</int>
      </lst>
    </lst>
    - <lst name="1000036">
      <int name="rankDir">0</int>
      <int name="fornelement">7</int>
    - <lst name="values">
      - <lst name="5260113">
        <int name="valueId">5260113</int>
        <str name="label">ABS Computer Technologies Inc.</str>
        <str name="rating">50</str>
        <int name="count">7</int>
      </lst>
    </lst>
  - <lst name="11795388">
    <int name="valueId">11795388</int>
  </lst>
Simple Faceted Request Handlers
SolrIndexSearcher s = req.getSearcher();
SolrQueryParser qp = new
    SolrQueryParser(req.getSchema(), null);
Query q = qp.parse( req.getQueryString() );

DocListAndSet results = s.getDocListAndSet
    (q, (List<Query>)null, (Sort)null,
     req.getStart(), req.getLimit());

NamedList counts = new NamedList();
    for (String fc : req.getParams("fc")) {
        counts.add(fc, s.numDocs(qp.parse(fc),
                results.docSet));
    }
    rsp.add("facet constraint counts", counts);
    rsp.add("your results", results.docList);

...
SimpleFacetedRequestHandler


- <response>
  - <responseHeader>
    <status>0</status>
    <QTime>1</QTime>
  </responseHeader>
  - <lst name="facet constraint counts">
    <int name="inStock:true">1</int>
    <int name="inStock:false">2</int>
  </lst>
- <result numFound="3" start="0">
  - <doc>
    - <arr name="cat">
      <str>electronics</str>
      <str>music</str>
    </arr>
    - <arr name="features">
      <str>iTunes, Podcasts, Audiobooks</str>
      <str>Stores up to 15,000 songs, 25,000 photos, or 150 hours of video</str>
      <str>2.5-inch, 320x240 color TFT LCD display with LED backlight</str>
      <str>Up to 20 hours of battery life</str>
      <str>Plays AAC, MP3, WAV, AIFF, Audible, Apple Lossless, H.264 video</str>
  </doc>
</result>
DynamicFacetedRequestHandler

IndexReader r = s.getReader();
NamedList facets = new NamedList();
for (String ff : req.getParams("ff")) {
    Map counts = new HashMap();
    facets.add(ff, counts);

    TermEnum te = r.terms(new Term(ff,""));
    do {
        Term t = te.term();
        if (null == t || ! t.field().equals(ff))
            break;

        counts.put(t.text(), s.numDocs
                    (new TermQuery(t), results.docSet));
    } while (te.next());
}

rsp.add("facet fields", facets);
rsp.add("my results", results.docList);
...
DynamicFacetedRequestHandler

?qt=dfacet&q=video&ff=cat&ff=inStock

- <lst name="facet fields">
  - <lst name="cat">
    <int name="search">0</int>
    <int name="memory">0</int>
    <int name="graphics">2</int>
    <int name="card">2</int>
    <int name="connector">0</int>
    <int name="software">0</int>
    <int name="electronics">3</int>
    <int name="copier">0</int>
    <int name="multifunction">0</int>
    <int name="camera">0</int>
    <int name="music">1</int>
    <int name="hard">0</int>
    <int name="scanner">0</int>
    <int name="monitor">0</int>
    <int name="drive">0</int>
    <int name="printer">0</int>
  </lst>
  - <lst name="inStock">
    <int name="F">2</int>
    <int name="T">1</int>
  </lst>
</lst>
In Conclusion...

Go Use Solr!
Faceted Searching
With Apache Solr

October 13, 2006
Chris Hostetter
hossman – apache – org
http://incubator.apache.org/solr/
What is Faceted Searching?
**Example: Epicurious.com**

### browse

**browsing by:** Beef

**refine by:** Courses | Dish | Cuisine | Season/Occasion | Special Considerations | Preparation

- Appetizers (53)
- Brunch (8)
- First Course (13)
- Main Course (923)
- Snacks (6)
- Breakfast (7)
- Desserts (2)
- Hors d’Oeuvres (26)
- Side (11)

<table>
<thead>
<tr>
<th>ranking</th>
<th>recipe name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>BRESADOLA CARPACCIO WITH GRILICHE VINAIGRETTE</td>
</tr>
<tr>
<td>4</td>
<td>SHORT STEAK WITH MARCOTS VERTS, CORN, AND PESTO</td>
</tr>
</tbody>
</table>

http://www.epicurious.com/recipes/find/browse/results?type=browse&att=82
Example: Nabble.com

Example: CNET.com

http://reviews.cnet.com/4566-6502_7-0.html
Aka: “Faceted Browsing”

"Interaction style where users filter a set of items by progressively selecting from only valid values of a faceted classification system"

- Keith Instone, SOASIS&T, July 8, 2004

Faceted Browsing - How User Interfaces Represent and Benefit from a Faceted Classification System
SOASIS&T, July 8, 2004
http://user-experience.org/uefiles/facetedbrowse/
http://user-experience.org/uefiles/facetedbrowse/KI-FB-SOASIST.pdf
Key Elements of Faceted Search

- No hierarchy of options is enforced
  - Users can apply facet constraints in any order
  - Users can remove facet constraints in any order

- No surprises
  - The user is only given facets and constraints that make sense in the context of the items they are looking at
  - The user always knows what to expect before they apply a constraint

Facets/Constraints available should make sense particularly constraints that have already been applied

User is probably shown a result count for a constraint in advance, but at a minimum they should never reach an empty result set
Explaining My Terms

• Facet: A distinct feature or aspect of a set of objects; “a way in which a resource can be classified”

• Constraint: A viable method of limiting a set of objects

Facets usually correspond to fields in your index
Constraints may be values, or complex queries

http://facetmap.com/glossary/ is source of quote ... they have a different term for “constraint” which i don't like as much.
Dynamic Taxonomy? No.

- Bad Description
- Taxonomy implies a hierarchy of subsets
  - Hierarchy implies ordered usage of constraints

http://www.searchtools.com/info/faceted-metadata.html
Why Is Faceted Searching Hard?

- **LOTS** of set intersections
- All permutations can't be easily precomputed

If you only allow the user to constrain one facet at a time, and in a particular order, then counting the objects that match each of the constraints for the “next” facet becomes relatively easy – ie...

```
select foo, count(*) where ... group by foo
```
What is Solr?
Elevator Pitch

"Solr is an open source enterprise search server based on the Lucene Java search library, with XML/HTTP APIs, caching, replication, and a web administration interface."
What Does That Mean?

- Information Retrieval application
- Java5 WebApp (WAR) with a web services-ish API
- Uses the Java Lucene search library
- Initially built at CNET
- Now an Apache Incubator project

Information Retrieval: The study of systems for indexing, searching, and recalling data, particularly text or other unstructured forms."

http://www.virtechseo.com/seoglossary.htm

“Information retrieval (IR) is the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hypertext networked databases such as the Internet or intranets, for text, sound, images or data.”

http://en.wikipedia.org/wiki/Information_retrieval
Lucene Refresher

- Lucene is a full-text search library
  - Maintains inverted index: terms -> documents
- Add documents to an index via IndexWriter object
  - A document is a collection of fields
  - No config files, dynamic field typing
  - Text analysis performed by Analyzer objects
  - No notion of "updating" or "replacing" an existing document
- Search for documents via IndexSearcher object
  \[ \text{Hits} = \text{search}(\text{Query}, \text{Filter}, \text{Sort}, \text{topN}) \]
- Scoring: \( \text{tf} \times \text{idf} \times \text{lengthNorm} \)
Solr in a Nutshell

- Index/Query via HTTP and XML
- Comprehensive HTML Administration Interfaces
- Scalability - Efficient Replication to Other Solr Search Servers
- Extensible Plugin Architecture
- Highly Configurable and User Extensible Caching
- Flexible and Adaptable with XML configuration
  - Data Schema with Dynamic Fields and Unique Keys
  - Analyzers Created at Runtime from Tokenizers and TokenFilters

http://incubator.apache.org/solr/features.html
Example: Adding a Document

HTTP POST /update

<add><doc>
  <field name="article">05991</field>
  <field name="title">Apache Solr</field>
  <field name="subject">An intro...</field>
  <field name="cat">search</field>
  <field name="cat">lucene</field>
  <field name="body">Solr is a full...</field>
  <field name="inStock">true</field>
</doc></add>

To replace an existing document with the same unique key (in this schema “article”) just re-add it

Adding documents requires a commit which opens a new IndexSearcher so the new documents are visible.
Example: Execute a Query

HTTP GET
/select/?qt=foo&wt=bar&start=0&rows=10&q=solr

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <responseHeader>
    <status>0</status><QTime>1</QTime>
  </responseHeader>
  <result numFound="1" start="0">
    <doc>
      <arr name="cat">
        <str>lucene</str><str>search</str>
      </arr>
      <bool name="inStock">true</bool>
      <str name="title">Apache Solr</str>
      <int name="popularity">10</int>
    </doc>
  </result>
</response>

QT is Query Type – which Request Handler will process the request

WT is Writer Type – which Response Writer will format the response

Neither option is required, default is “standard”
Example: SimpleRequestHandler

```java
public void handleRequest(SolrQueryRequest req,
                          SolrQueryResponse rsp) {
  try {
    Query q = QueryParsing.parseQuery
               (req.getQueryString(), req.getSchema());

    DocList results =
               req.getSearcher().getDocList
               (q, (Query)null, (Sort)null,
                req.getStart(), req.getLimit());

    rsp.add("simple results", results);
    rsp.add("other data", new Integer(42));
  } catch (Exception e) {
    rsp.setException(e);
  }
```

NOTE: To save space, the class declaration, and some other basic methods defined in the SolrRequestHandler interface have been omitted.

This method illustrates the basics of what StandardRequestHandler does -- minus statistics, debugging, highlighting, field selection, etc...

QueryParsing.parseQuery uses a SolrQueryParser which is aware of the schema.xml and can apply the appropriate Analyzer to each field used.

In addition to DocLists any "simple type" can be added to the response...

- Null
- String
- Integer, Long
- Float, Double
- Date
- Boolean
- Collection or Array of "simple type"
- Map or NamedList of String => "simple type"
DocLists and DocSets

- **DocList** - An ordered list of document ids with optional score
  - A subset of the complete list of documents actually matched by a Query

- **DocSet** - An unordered set of Lucene Document Ids
  - Typically the complete set of documents matched by a query
  - Multiple implementations optimized for different size sets
  - Foundation of Faceted Searching in Solr

Two implementations of DocSet allow for optimizations based on size of set.

HashDocSet used for small sets, OpenBitSet based BitDocSet used for larger sets.

(OpenBitSet is 3 to 4 times faster than java.util.BitSet for set intersections)
Caching

- IndexSearcher's view of an index is fixed
  - Aggressive caching possible
  - Consistency for multi-query requests

- Types of Caches:
  - filterCache: Query => DocSet
  - resultCache: (Query, Sort, Filter) => DocList
  - documentCache: docId => Document
  - userCaches: Object => Object
  - application specific, custom query handlers
• Static warming requests configured from solrconfig.xml, triggered by events (newSearcher or firstSearcher)

• Autowarming: The top keys from the old (current) cache are re-queried using the new IndexSearcher to pre-populate the new cache(s).

• Cache specific regenerators are used that take keys from old caches and use the new Searcher to pre-populate the new caches.

• The docCache does not have autowarming done since document ids change from one searcher to the next.

• Lucene also has some internal caches (FieldCache and field norms) than benefit from warming.

• After all warming is completed, the new IndexSearcher is registered, and starts serving live requests

• The old index searcher hangs around until all of it’s requests have completed, then it is closed.
Case Study

CNET's First
Solr Powered Page
Static Pulldowns, many permutations lead to dead pages; Even if you selected one at a time the next page would still list all options for all pulldowns, giving you more options for blank pages
### Shiny New Faceted Version

**Find by price**
- $50 - $150 (10)
- $155 - $250 (20)
- $255 - $320 (15)
- $325 - $450 (15)
- $450 - $600 (17)
- See all prices

**Find by manufacturer**
- Ais Communications (42)
- Logitech Inc. (44)
- 42EM Corporation (21)
- Panasonic (19)
- Creative Labs Inc. (18)
- See all manufacturers

**Find by audio input type**
- Microphone (94)
- None (92)
- Headset (7)

**Or find by**
- Compatibility
- Connector type
- Interface type

---

**Microsoft LifeCam VX-6000**

The Microsoft LifeCam VX-6000 offers unique features such as the ability to post photos directly to a blog, but its video effects and image quality don't stand up to that of competing Webcams from veteran manufacturers.

**Specs:** Drivers & Utilities

- Add to my products
- New! What is this?

**Creative Live Cam Voice**

With beefed-up audio features, the Creative Live Cam Voice is one of the best overall cameras for IM and Internet-based voice-conferencing.

**$70 to $99**

- at 4 stores

**$74 to $89**

- at 3 stores

---

**List of Facets is category specific**

**Constraints are category specific even if the facet is reused in multiple categories**

**Metadata determines display of constraints**

---

http://reviews.cnet.com/4566-6502_7-0.html
Category Metadata

- Category ID and Label
- Category Query
- Ordered List of Facets
  - Facet ID and Label
  - Facet "Display Type"
  - Ordered List of Constraints
    - Constraint ID and Label
    - Constraint Query

US 2006
Key Features We Needed In Solr

- Loose Schema with Dynamic Fields
- Efficient implementation of sets and set intersection
- Aggressive set caching
- Plugin Architecture

Dynamic Fields – for storing different fields for different types of products
Plugins – for putting our biz logic in the Solr server so we wouldn’t need to stream all of the set data to our application
We store our Category metadata in Solr Documents with different fields from our product Documents. (Mainly because that way Solr takes care of replication to our slaves).

getFirstMatch is a helper method for getting the first document matching a query – useful when you know the uniqueKey of a document you want.

parseAndCacheMetadata utilizes a Solr userCache to store the Metadata objects keyed off of the category Id.

getDocListAndSet is an optimized way to retrieve both the DocSet of all matches as well as the DocList for the current Sort/pagination – it caches both automatically.

SolrIndexSearcher.numDocs is a convenience method that finds the intersection of two Queries (or a Query an a DocSet). It currently just fetches the DocSets from each, using the filterCache, but in the future it may use its own cache of (Query, Query) => Integer for a more memory efficient lookup of common intersections.
Conceptual Picture

getDocListAndSet(Query, Query[], Sort, offset, n)

computer_type: PC
memory: [1GB TO *]
price asc

Section of ordered results

DocList

DocSet

Unordered set of all results

numDocs()

proc_manu: Intel = 594
proc_manu: AMD = 382

price: [0 TO 500] = 247
price: [500 TO 1000] = 689

manu: Dell = 104
manu: HP = 92
manu: Lenovo = 75
XML Response

- <response>
  - <responseHeader>
    - <status>0</status>
    - <QTime>17</QTime>
  </responseHeader>
  + <result name="products" numFound="5481" start="0">
    - <list name="metadata">
      - <int name="rankDir">1</int>
      - <int name="formelement">10</int>
      + <list name="values">
        - <int name="datatype">3</int>
        - <int name="rating">54</int>
        - <str name="name">Price</str>
        - <int name="attributed">100021</int>
      </list>
    </list>
    - <list name="1000036">
      - <int name="rankDir">0</int>
      - <int name="formelement">7</int>
      + <list name="values">
        - <int name="valueid">5260113</int>
        - <str name="label">ABS Computer Technologies Inc.</str>
        - <str name="rating">50</str>
        - <int name="count">7</int>
      </list>
    </list>
    - <list name="11795388">
      - <int name="valueid">11795388</int>
  </list>
</result>
Simple Faceted Request Handlers
SimpleFacetedRequestHandler

```java
SolrIndexSearcher s = req.getSearcher();
SolrQueryParser qp = new
    SolrQueryParser(req.getSchema(), null);
Query q = qp.parse( req.getQueryString() );

DocListAndSet results = s.getDocListAndSet
    (q, (List<Query>)null, (Sort)null,
     req.getStart(), req.getLimit());

NamedList counts = new NamedList();
    for (String fc : req.getParams("fc")) {
        counts.add(fc, s.numDocs(qp.parse(fc),
                         results.docSet));
    }
    rsp.add("facet constraint counts", counts);
    rsp.add("your results", results.docList);
```

NOTE: To save space, the method declaration and basic Exception handling already seen in the SimpleRequestHandler have been left out.

Facet Constraints are being specified via request params – they could just as easily be coming from init params or a separate config file.

List<Query> is where any constraints the user has selected would be applied – they are evaluated independently from the main query so:

• they don’t affect scoring
• they leverage the DocSet cache (which should be a cache hit from earlier requests when the facet constraint counts were generated)

CAVEAT: As shown, this code is error prone (In particular, the for loop can result in an NPE if no “fc” params are specified). A well written RequestHandler would do more robust param validation and error checking.
SimpleFacetedRequestHandler

?q facet=video&fc=inStock:true&fc=InStock:false

- <response>
  - <responseHeader>
    <status>200</status>
    <QTime>1</QTime>
  </responseHeader>
  - <lst name="facet constraint counts">
    <int name="inStock:true">1</int>
    <int name="inStock:false">2</int>
  </lst>
  - <lst name="facet">
    <int>
      <int>
      </int>
    </int>
  </lst>
  - <lst name="facet">
    <int>
      <int>
      </int>
    </int>
  </lst>
  - <arr name="cat">
    <str>electronics</str>
    <str>music</str>
  </arr>
  - <arr name="controls">
    <str>Tunes, Podcasts, Audiobooks</str>
    <str>Stores up to 15,000 songs, 25,000 photos, or 150 hours of video</str>
    <str>2.5-inch, 320x240 color TFT LCD display with LED backlight</str>
    <str>Up to 20 hours of battery life</str>
    <str>Plays AAC, MP3, WAV, AIFF, Audible, Apple Lossless, H.264 video</str>
  </arr>
</response>
DynamicFacetedRequestHandler

IndexReader r = s.getReader();
NamedList facets = new NamedList();
for (String ff : req.getParams("ff")) {
    Map counts = new HashMap();
    facets.add(ff, counts);

    TermEnum te = r.terms(new Term(ff,""));
    do {
        Term t = te.term();
        if (null == t || ! t.field().equals(ff))
            break;

        counts.put(t.text(), s.numDocs
            (new TermQuery(t), results.docSet));
    } while (te.next());
}

rsp.add("facet fields", facets);
rsp.add("my results", results.docList);

NOTE: To save space, the method declaration, basic Exception handling, and basic query execution already seen in the SimpleRequestHandler and SimpleFacetedRequestHandler have been left out.

Facet Fields are being specified via request params – they could just as easily be coming from init params or a separate config file.

SolrIndexSearcher.getReader exposes the low level Lucene IndexReader that Solr Is using for RequestHandlers that want to do low level things.

TermEnum is a low level Lucene class that allows direct access to the list of all terms in the index, with fast methods to skip ahead to the lexigraphically “lowest” existing term after a specified term.

The key difference between this RequestHandler and the previous one, is that the constraints themselves are being driven by the data in the index.

CAVEAT: As shown, this code is error prone (In particular, the TermEnum is a tricky beast which may be null, or may return terms which are null. Also: This code is dealing with the raw term text, which for some Solr field types may be encoded and not human readable). A well written RequestHandler would do more robust param validation and error checking.
DynamicFacetedRequestHandler

?qt=dfacet&q=video&ff=cat&ff=inStock

  <lst name="facet fluids">
  <int name="cat">
    <int name="search">0</int>
    <int name="memory">0</int>
    <int name="graphics">2</int>
    <int name="card">2</int>
    <int name="connector">0</int>
    <int name="software">0</int>
    <int name="electronics">3</int>
    <int name="copier">0</int>
    <int name="multifunction">0</int>
    <int name="camera">0</int>
    <int name="music">1</int>
    <int name="hard">0</int>
    <int name="scanner">0</int>
    <int name="monitor">0</int>
    <int name="drive">0</int>
    <int name="printer">0</int>
  </int>
  <int name="inStock">
    <int name="F">2</int>
    <int name="T">1</int>
  </int>
</lst>
In Conclusion...

Go Use Solr!