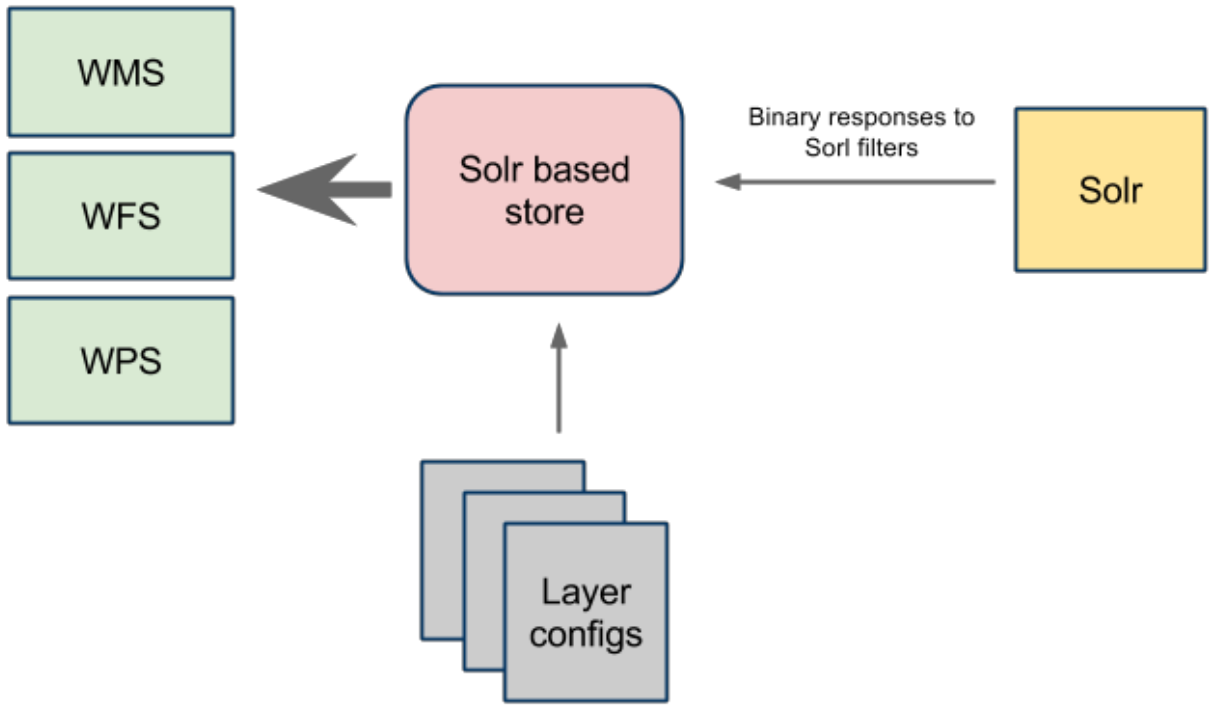
SORL based data store

The document is a specification for a GeoTools DataStore that has the ability to connect to a SOLR server and extract simple features out of it, to be presented as layers, allowing efficient rendering of hundreds of Sorl spatial documents.



Also, we want to be able to pass down SOLR request parameters as GeoServer view params, using the same infrastructure as sql views (same request parameters, same query hints to vehiculate them down into the store):
http://docs.geoserver.org/stable/en/user/data/database/sqlview.html#parameterizing-sql-views

Here is a sample request and response to SOLR:

http://localhost:8983/solr/select/?indent=on&q=video&fl=*,score

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">1</int>
    <lst name="params">
      <str name="indent">on</str>
      <str name="q">video</str>
```

```
      <str name="fl">*,score</str>
    </lst>
  </lst>
  <result name="response" numFound="3" start="0" maxScore="0.39733005">
    <doc>
      <float name="score">0.39733005</float>
      <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>
        <str>Notes, Calendar, Phone book, Hold button, Date display,
          Photo
          wallet, Built-in games, JPEG photo playback, Upgradeable firmware,
          USB 2.0 compatibility, Playback speed control, Rechargeable
          capability, Battery level indication</str>
      </arr>
      <str name="id">MA147LL/A</str>
      <bool name="inStock">true</bool>
      <str name="includes">earbud headphones, USB cable</str>
      <str name="manu">Apple Computer Inc.</str>
      <date name="manufacturedate_dt">2005-10-12T08:00:00Z</date>
      <str name="name">Apple 60 GB iPod with Video Playback Black</str>
      <int name="popularity">10</int>
      <float name="price">399.0</float>
      <str name="store">37.7752,-100.0232</str>
      <float name="weight">5.5</float>
    </doc>
  </result>
</response>
```

Issues and features:

- Solr information is a set of **unstructured documents** that can contain a well known set of typed attributes, but does not have a attribute grouping concept (object, table)
- Solr data can contain dynamic, unforeseen attributes
- Solr attributes can be **multivalued**
- Solr data can have a spatial component in the form of a geometry, we want to leverage it
- All OGC filtering information should be passed down to Solr as the query for fast data retrivial, translating them to f and fq clauses, and merging them with the eventual extra f and fq clauses coming from the view params

Configuration UI and internal machinery

Store configuration

The data store should be configured with only two properties:

- A URL to connect to the Solr instance, eg., http://localhost:8983/solr
- The field used in Solr that holds the layer names, e..g, “layerType”, the unique values of this field will represent the layer names and the basic form of filtering to tell apart documents associated to different layers (the GUI will do a call to SOLR to determine the domain)

New layer creation and confguration issues

When creating a new layer the GUI will list the available layer names as usual, once chosen the GUI will have to assist with the configuration of the layer, which will start with no attributes, by presenting:

- ability to add/remove/reorganize attributes coming from the Solr column definitions
- identify a column as holding the geometry
- for simplicity at the moment we assume multi-valued properties will get only the first value in the output, so no configuration desired here

The above can be implemented by adding a new ResourceConfigurationPanelInfo/ResourceConfigurationPanel that will activate only if the layer is coming from a Solr source.

The configuration bean (e.g., SolrLayerConfiguration) is to be added into the FeatureTypeInfo as a metadata map entry (there is a need to roll a custom XStreamPersisterInitializer to handle it, see the dyndimension GeoServer community module as an example)

This information will be then used in ResourcePool to configure the layer. To do this in a pluggable manner, we need to roll a new pluggable extension that allows the ResourcePool to delegate the feature type initialization to.

It would be a generalization of what we are currently doing for JDBCDataStores in ResourcePool:

https://github.com/geoserver/geoserver/blob/master/src/main/src/main/java/org/geoserver/catalog/ResourcePool.java#L872
https://github.com/geoserver/geoserver/blob/master/src/main/src/main/java/org/geoserver/catalog/ResourcePool.java#L844
https://github.com/geoserver/geoserver/blob/master/src/main/src/main/java/org/geoserver/catalog/ResourcePool.java#L1112

That is, create a FeatureTypeInitializer interface that we can use to delegate all types of special initializations, the interface would act as a broker between the programmatic interfaces provided

by GeoTools and the configuration stored in the feature type metadata maps.
This requires a GSIP in the GeoServer community.

Data filtering

WMS/WFS request often come with some sort of filters, like non spatial property comparisons. The filters will be encoded as well as possible into Solr equivalents and sent down to the Solr server to reduce the amount of data to be read.
It is however recognized that a 1-1 encoding might not be possible, in that case in memory post filtering will be used to ensure only the data fitting the OGC filter is actually returned to the user, like in the JDBCDataStore:
https://github.com/geotools/geotools/blob/master/modules/library/jdbc/src/main/java/org/geotools/jdbc/JDBCFeatureSource.java#L566

In addition, any parameter coming down in the viewParams query hint will be added to the request, in case of overlap with the filters present in the Query, they will be AND-ed to them.

Efficient/memory conscious data transfer

Solr can transfer data in many ways, for example, Json, XML and so on. The most efficient protocol available so far is the binary one (http://wiki.apache.org/solr/javabin), which the SolrJ library (http://wiki.apache.org/solr/Solrj) uses to communicate with Solr.

Using the library is tempting, but it has a major drawback: it parses all the result into an in memory collection. Given the main point of the exercise is to render maps with hundreds of thousands of geometies, we have to use the binary protocol, but in such a way that we can stream through it, reading one featuer at a time from the binary stream instead of parsing everything in advance.
So, a streaming binary parser for the Solr binary protocol will have to be implemented, which will likely use the JavaBinCode class to do most of the work
http://svn.apache.org/viewvc/lucene/dev/trunk/solr/solrj/src/java/org/apache/solr/common/util/JavaBinCodec.java?view=markup