MySolr

:TODO: Work in progress - Help wanted

⚠️ :TODO: ⚠️ This document is not fully finished (since there are still a couple of open questions) but it may be useful for people getting started with solr development based on Ant. Please share your experience, collaborating on this document while you test the above. Please use our mailing lists for all communications.

⚠️ :TODO: ⚠️ It has also been pointed out that this document is a little misleading about how to setup and run solr - particularly given that it has it's own ant build.xml file embedded in it for the purpose of doing a lot of things that aren't relaly neccessary for most users downloading Solr releases (ie: recompiling solr).

⚠️ :TODO: ⚠️ This doc is very confusing in it's use of "SOLR_HOME" as an environment variable referring the directory where the Solr code base (including build.xml and source code) is checked out – this is inconsistent with standard SolrTerminology.

Introduction

Solr is an open source enterprise search server based on the Lucene Java search library. Solr tries to expose all of the Lucene goodness through an easy to use, easy to configre, HTTP interface. Besides the configuration, Solr's other means of being a value add is in it's IndexReader management, it's caching, and it's plugin support for mixing and matching request handlers, output writers, and field types as easily as you can mix and match Analyzers.

Architecture

Production

Typically it's not recommended to have your front end users/clients hitting Solr directly as part of an HTML form submit ... the more conventional way to think of it is that Solr is a backend service, which your application can talk to over HTTP – if you were dealing with a database, you wouldn't expect that you could generate an HTML form for your clients and then have them submit that form in some way that resulted in their browser using JDBC (or ODBC) to communicate directly with your database, their client would communicate with your App, which would validate their input, impose some security checks on the input, and then execute the underlying query to your database – working with Solr should be very similar, it just so happens that instead of using JDBC or some other binary protocol, Solr uses HTTP, and you *can* talk to it directly from a web browser, but that's really more of a debugging feature then anything else.

Development

This document describes how to create and deploy your custom solr server based on an out-of-the-box solr distribution. We will use Apache Ant for the deployment and possible patching of standard solr files.

Getting started

If you followed the tutorial you may now ask yourself how to create your own application based on SOLR. We use the example webapp dir structure and strip everything that we do not need or do not want to duplicate. The most likely files you need are in the solr/conf directory since they are controlling the behavior of our application.

First we will setup a basic directory structure (assuming we only want to change some fields) and copy the attached build.xml and solr.xml:

```
$mySolr
|-- solr
|   `-- conf
|       |-- schema.xml
|       `-- solrconfig.xml
|-- solr.xml
`-- build.xml
```

In the build.xml we will later generate a fully functional web-application using Jetty as servlet engine. The following is the basic structure when you are developing with a standalone jetty included. Where etc/, ext/, lib/ and start.jar are jetty specific and not needed when developing for tomcat. However in our small example we will use jetty and copy this files from the example via our ant script.

The most important directoy is solr/. Here we store all configuration parameter needed by solr where the idea is our above defined files have priority before the core files from solr which we will copy over via our little ant script.

Setting up your own fields

As soon you are using your own fields you will have to define them in the schema.xml. Since we used the example schema as basis for our app, remove the following fields and add your own:

```
<field name="sku" type="textTight" indexed="true" stored="true" omitNorms="true"/>
<field name="name" type="text" indexed="true" stored="true"/>
<field name="manu" type="text" indexed="true" stored="true" omitNorms="true"/>
<field name="cat" type="text_ws" indexed="true" stored="true" multiValued="true" omitNorms="true"/>
<field name="features" type="text" indexed="true" stored="true" multiValued="true"/>
<field name="includes" type="text" indexed="true" stored="true"/>

<field name="weight" type="sfloat" indexed="true" stored="true"/>
<field name="price"  type="sfloat" indexed="true" stored="true"/>
<field name="popularity" type="sint" indexed="true" stored="true"/>
<field name="inStock" type="boolean" indexed="true" stored="true"/>
<field name="manu_exact" type="string" indexed="true" stored="false"/>
```

Ant script

The following script is taken from an application where solr is one part of a selection of other components. It is thought of being imported by the application main build.xml. You need at least Apache Ant version 1.7.0 since we are using prefixing of the target names which have been introduced in 1.7.

```
<?xml version="1.0"?>
<project>
  <!-- SOLR_HOME must be set as an environment variable -->
  <property name="solr.home" value="${env.SOLR_HOME}"/>
  <condition property="solr.set">
    <isset property="solr.home"/>
  </condition>
  <fail unless="solr.set">Please set SOLR_HOME in your environment. export
    SOLR_HOME=~/src/apache/solr/ </fail>
  <import file="${solr.home}/build.xml"/>
  <property name="solr.build" value="${build}/solr"/>
  <property name="solr.src" value="${solr.home}/src"/>
  <target name="mysolr.compile">
    <!-- The runtime classpath -->
    <path id="compile.classpath.solr">
      <path refid="ant.classpath"/>
      <fileset dir="${lib}">
        <include name="*.jar"/>
      </fileset>
      <fileset dir="${solr.home}/lib">
        <include name="*.jar"/>
      </fileset>
      <pathelement location="${solr.build}/common"/>
    </path>
    <echo>dest: ${solr.build}/common</echo>
    <echo>src: ${solr.src}/java</echo>
    <solr-javac destdir="${solr.build}/common"
      classpathref="compile.classpath.solr">
      <src path="${solr.src}/java"/>
      <include name="org/apache/solr/common/**"/>
    </solr-javac>
    <solr-javac destdir="${solr.build}/core"
      classpathref="compile.classpath.solr">
      <src path="${solr.src}/java"/>
      <src path="${solr.src}/webapp/src"/>
      <exclude name="org/apache/solr/common/**"/>
    </solr-javac>
  </target>
  <target name="mysolr.dist-jar" depends="mysolr.compile, solr.make-manifest">
    <mkdir dir="${dist}" />
    <solr-jar
         destfile="${dist}/${fullnamever}-solr.jar"
         basedir="${solr.build}/core" />
    <!-- package the common classes together -->
    <solr-jar
         destfile="${dist}/${fullnamever}-solr-common.jar"
```

```
        basedir="${solr.build}/common" />
  </target>
  <target name="mysolr.dist-war" depends="mysolr.dist-jar">
    <war destfile="${dist}/solr.war"
         webxml="${solr.src}/webapp/WEB-INF/web.xml"
         filesetmanifest="skip"
         manifest="${build}/META-INF/MANIFEST.MF">
      <lib dir="${solr.home}/lib">
        <exclude name="servlet-api*.jar" />
        <exclude name="easymock.jar" />
      </lib>
      <lib dir="${dist}">
        <include name="${fullnamever}-solr.jar" />
        <include name="${fullnamever}-solr-common.jar" />
      </lib>
      <fileset dir="${solr.src}/webapp/resources" />
    </war>
  </target>
  <target name="mysolr.dist" depends="mysolr.dist-war">
    <property name="solr.app" value="${dist}/solrappp"/>
    <property name="sorl.example" value="${solr.home}/example"/>
    <mkdir dir="${solr.app}"/>
    <mkdir dir="${solr.app}/logs"/>
    <copy todir="${solr.app}">
      <fileset dir="${boja.solr}" includes="**,*"/>
    </copy>
    <copy todir="${solr.app}/etc">
      <fileset dir="${sorl.example}/etc" includes="**,*"/>
    </copy>
    <copy todir="${solr.app}/lib">
      <fileset dir="${sorl.example}/lib" includes="**,*"/>
    </copy>
    <copy todir="${solr.app}/solr/bin">
      <fileset dir="${solr.src}/scripts">
        <exclude name="scripts.conf"/>
      </fileset>
    </copy>
    <chmod dir="${solr.app}/solr/bin" perm="755" includes="**"/>
    <copy file="${dist}/solr.war" tofile="${solr.app}/webapps/solr.war"/>
    <copy todir="${solr.app}/solr/conf" overwrite="false">
      <fileset dir="${sorl.example}/solr/conf" includes="**,*"/>
    </copy>
    <chmod dir="${solr.app}/solr/conf" perm="755" includes="schema.xml,solrconfig.xml"/>
    <copy file="${sorl.example}/start.jar" tofile="${solr.app}/start.jar"/>
    <echo>Solr successfully builded.</echo>
    <echo>You can excute the following command to start the server:Solr successfully builded.</echo>
    <echo>cd ${solr.app};java -jar start.jar</echo>
  </target>
</project>
```

Now you can execute `ant mysolr.dist` and it will generate your new mySolr server.

Basic structure with jetty after build

```
$mySolr
|-- etc
|   |-- jetty.xml
|   `-- webdefault.xml
|-- lib
|   |-- jetty-6.1.3.jar
|   |-- jetty-util-6.1.3.jar
|   |-- jsp-2.1
|   |   |-- ant-1.6.5.jar
|   |   |-- core-3.1.1.jar
|   |   |-- jsp-2.1.jar
|   |   `-- jsp-api-2.1.jar
|   `-- servlet-api-2.5-6.1.3.jar
|-- logs
|   `-- 2007_09_07.request.log
|-- solr
|   |-- bin
|   |   |-- abc
|   |   |-- abo
|   |   |-- backup
|   |   |-- backupcleaner
|   |   |-- commit
|   |   |-- optimize
|   |   |-- readercycle
|   |   |-- rsyncd-disable
|   |   |-- rsyncd-enable
|   |   |-- rsyncd-start
|   |   |-- rsyncd-stop
|   |   |-- scripts-util
|   |   |-- snapcleaner
|   |   |-- snapinstaller
|   |   |-- snappuller
|   |   |-- snappuller-disable
|   |   |-- snappuller-enable
|   |   `-- snapshooter
|   |-- conf
|   |   |-- admin-extra.html
|   |   |-- protwords.txt
|   |   |-- schema.xml
|   |   |-- scripts.conf
|   |   |-- solrconfig.xml
|   |   |-- stopwords.txt
|   |   |-- synonyms.txt
|   |   `-- xslt
|   |       |-- example.xsl
|   |       |-- example_atom.xsl
|   |       |-- example_rss.xsl
|   |       `-- luke.xsl
|   `-- data
|       `-- index
|           |-- segments.gen
|           `-- segments_1
|-- start.jar
`-- webapps
    `-- solr.war
```

Add your documents to the index

Add your own documents (conform to the schema definitions you used) as mentioned in the tutorial and test.

Open questions that you may have

1. The schema.xml and solrconfig.xml are in parts very well explained. But in some areas like as an example .. <indexDefaults> and other places there are no explanation. It would be nice to get more info there. Specifically for example if increase <mergeFactor> to 1000 what will happen? what are the highest value for each properties? what is for example a "safe value". 2. It would be nice to create a deployment scenarios i.e a

single server install with XXX CPU and YYY memory just running Solr with AAA thousand docs how should your config look like and why? and you can get about xxx Query/Sec or something.. 3. It would be nice to have a multi server deployment with some server spec and then how should the deployment be. 4. It would also be nice to have more info regarding stopwords synonoms etc. usage and facet etc..