Marc's Nutch 0.7.1 Page

My Nutch Oddessy - Getting Nutch 0.7.1 up and running

A litte history. This is a little information on how I use Nutch. I'm not a Nutch developer and there's no indepth commentary on the architecture or roadmap for Nutch here. It's just my attempt to convey some of what I've learned and done over the past couple of weeks with Nutch 0.7.1 and hopefully it'll help some other folks out there who are trying to get Nutch up and running for their own purposes.

I set out to try and put into place a mechanism to search the content of an intranet. The content of this intranet contains a number of web pages, but more importantly it contains documentation. These documents consist of MS word, MS powerpoint, MS excel, PDF, text, html, and other file types, some of which have no Nutch plugin currently. After a few attempts to cobble something together with things like glimpse, perl, shell scripts, etc. I came to the realization that while those solutions might work, they really didn't work well, were going to be difficult to maintain, and couldn't be extended easily. What I began to realize was that I needed a web crawler. Hence my quest began and eventually led me to Nutch... perfect timing. Nutch 0.7.1 had only been released some 5 days prior to my search. After reading through the Wiki, email threads, forums, etc. I decided that Nutch was the way to go... so I started researching what software I needed to download and how I might go about getting it up and running for my purposes; crawling an intranet with the objective of providing documentation searching based upon an in-depth scan and indexing of those documents. So here's where I began...

Tools

*apache-ant-1.6.5-bin.tar.gz
*j2sdk-1_4_2_09-linux-i586-rpm.bin
*jakarta-tomcat-4.1.31.tar.gz
*nutch-0.7.1.tar.gz
*poi-bin-2.5.1-final-20040804.tar.gz
*poi-src-2.5.1-final-20040804.tar.gz

Plugins

*http://issues.apache.org/jira/browse/NUTCH-52?page=all
*http://issues.apache.org/jira/browse/NUTCH-21?page=all

So, getting Nutch to just crawl was straightforward. I just followed the basic crawl tutorial like everyone else and I actually had it crawling to a limited depth and breadth and parsing pdf and doc files in no time. But...

Then you need to scale up, want to add file types, and perhaps need a few changes, like HTTP basic authentication, which requires some reconfiguration and no matter how you slice it also requires some tweaking of the source and a recompile or two (or 20 or 30 if you're me) 😉

So here's what I did and a little why, not necessarily in any particular order.

Configuration:

file: nutch-site.xml

***http.timeout** - I set this to 100000 because I have to deal with access to a clearcase based document repository and it can be sloooow.

***http.max.delays** - I also set this to 100000 for the same reason. There's only one host and it can be slow.

***fetcher.server.delay** - I set this to 0.1. Even though there's one host I don't want the fetcher threads sitting around all day before they start to fetch the next URL. Setting this lower drops latency between fetches, over time those can add up.

***fetcher.therads.fetch** - I set this to 15. There are 3 hosts that my crawl would access and I only wanted a max of 5 threads per host (see below). I'm not sure what kind of parallelism can be achieved with these threads on a single CPU host and I'm not willing to spend the time to really investigate further. Let's just say I feel good with these at 15.

***fetcher.threads.per.host** - I set this to 5.

***parser.threads.parse** - I set this to 15 in line with the 15/5 ratios that I've setup for my purposes

***indexer.max.title.length** - I set this to 1024. We have document naming conventions that can produce some really long names.

***indexer.max.tokens** - I set this to 10000. Although we do have some really large documents the little PC that I've been using to crawl doesn't have much memory (physical and virtual) so I'll have to live with any resulting truncation until I get more powerful HW.

***ftp.keep.connection** - I set this to true. Given that I'm only accessing a few hosts and accessing them over and over it doesn't make sense to close those connections as that just creates more latency when crawling.

***http.content.limit** - I set this to -1. This allows for full downloading of the file. I am guessing that any further size restrictions are then imposed by indexer.max.tokens (without peeking into the source to verify this).

***plugin.includes** - I updated the regex to include pdf|msword|powerpoint

*__http.auth.basic.username__ - This is a bit special as it is part of my HTTP basic authentication hack. The value of this would be your userid. More on this below.

*__http.auth.basic.password__ - Again part of the HTTP basic authentication hack. The value of this would be your password. All of you IS admins cringing; I know, not secure, but it works.

*__http.auth.verbose__ - I set this to true so that some additional debugging would be available in the logs.

*__fetcher.verbose__ - I set this to true for debugging.

*__http.verbose__ - I set this to true for debugging

*__parser.html.impl__ - I set this to tagsoup. Some of the pages that I was parsing did not adhere to strict HTML syntax and subsequently some of the URLs that I was wanting to be included in the fetch were being discarded. Using tagsoup let me catch those URLs and prune out others as long as my regex sets were a bit more specific.

---

file: crawl-urlfilter.txt

Just recall that the regexes are evaluated in top down order. So if you want something discarded early it needs to go higher up in the file. My extension pruning regular expression has gotten a little big at this point but there is a ton of content there and I really don't want this stuff in the mix.

```
-\.
(asm|bac|bak|bin|c|cat|cc|cdf|cdl|cfg|cgi|cpp|css|csv|dot|eps|exe|fm|gif|GIF|gz|h|ics|ico|ICO|iso|jar|java|jpg|JPG|l|lnt|mdl|mif|mov|MOV|mpg|mpp|msg|mso|mspat|mtdf|ndd|o|oft|orig|out|pjt|pl|pm|png|PNG|prc|prp|ps|rpm|rtf|sh|sit|st|tar|tb|tc|tgz|wmf|xla|xls|xml|y|Z|zip)$
```

Then I prune away based on host name.

```
-^http://hostthatidontwant.my.domain.name/.*
```

Then I prune away paths that are in hosts that I do want, e.g. don't include any doxygen pages for source browsing

```
-.*/doxygen/.*
```

Then put in the hosts that I do want starting at the depth that I want in certain scenarios

```
+^http://hostthatiwant.my.domain.com/.*
+^http://anotherhostiwant.my.domain.com/and/start/here/.*
```

Then skip everything else

```
-.
```

That's most of the configuration that I did for crawling, recalling that I'm crawling my own pages and am particularly interested in deep inspection of the documents that I find so that I can search them.

Compilation

This is where things started to get to be fun. I'll just list out what I was wanting to do and what I did to resolve the issues that I came across so that you might be able to do the same.

Compiling out of the box

This doesn't seem to work with 0.7.1. Given that this is still a 0.x.x release I'm not completely surprised, but then again you'd expect that a release would compile straight away just the same.

The first issue I had was with the

```
 nutch-extensionpoints
```

plugin directory. It fails to compile unless there is a **`src/java`** subdirectory. For some reason these directories aren't in the main branch or dev branches, but if you just add them ant will stop complaining and move along with nothing more to see.

```
$NUTCH_BASE/nutch-0.7.1/src/plugin/nutch-extensionpoints/src/java
```

add those last two dirs and ant will stop complaining.

Hacking in Basic Authentication

Reading through the source one gets a sense of where the development of authentication might be headed, but I still needed a quick hack to get basic authentication up and going. There's something in place for NTML, but nothing for just basic authentication. So just following the existing NTML implementation as a paradigm, knowing that it will disappear eventually, I added the code below. I may get around to adding patch files to this page for things that I did at some point.

file: nutch-site.xml

Earlier I mentioned this snip of xml that I added due to the HTTP basic authentication hack... it looks like this:

```
<property>
  <name>http.auth.basic.username</name>
  <value>myuserid</value>
  <description> HTTP authentication
  </description>
</property>

<property>
  <name>http.auth.basic.password</name>
  <value>hisign</value>
  <description> HTTP authentication
  </description>
</property>
```

Then reading through the java docs I came across the UsernamePasswordCredentials class which is specifically for capturing data for basic authentication.

file: nutch-0.7.1/src/plugin/protocol-httpclient/src/java/org/apache/nutch/protocol/httpclient/Http.java

```
import org.apache.commons.httpclient.UsernamePasswordCredentials;
```

Inside the Protocol implementation,

```
public class Http implements org.apache.nutch.protocol.Protocol {
...
static String BASIC_USERNAME = NutchConf.get().get("http.auth.basic.username", "");
static String BASIC_PASSWORD = NutchConf.get().get("http.auth.basic.password", "");
...
    if (NTLM_USERNAME.length() > 0) {
      Credentials ntCreds = new NTCredentials(NTLM_USERNAME,
                                              NTLM_PASSWORD,
                                              NTLM_HOST, NTLM_DOMAIN);
      client.getState().setCredentials(new AuthScope(NTLM_HOST,
                                                     AuthScope.ANY_PORT),
                                       ntCreds);
      LOG.info("Added NTLM credentials for " + NTLM_USERNAME);
    } else if (BASIC_USERNAME.length() > 0) {
     client.getState().setCredentials(new AuthScope(AuthScope.ANY_HOST,
                                                    AuthScope.ANY_PORT),
                                      new UsernamePasswordCredentials(
                                                    BASIC_USERNAME,
                                                    BASIC_PASSWORD));
      LOG.info("Added BASIC credentials for " + BASIC_USERNAME);
    }
...
```

What you should immediately notice is that the AuthScope object allows ANY_HOST and ANY_PORT and applies the credentials uniformly across them. Given that I'm crawling an intranet I really don't have to concern myself with the user/pass changing so I didn't add anything for flexibility there. You may want to do this should you need to supply different credentials for different hosts in your network.

MS Powerpoint plugin failing

So then I recompile and basic authentication starts working and I can access my documentation reposity as I wanted... WOOT! Okay, this was not without a good deal of trial and error on my part in all honesty. But, alas other things start to seep in. The powerpoint plugin starts to fail due to the mime type not matching the regex appropriately. You can search the Nutch email archive for vnd.ms-powerpoint and you'll find some discussion of this. To get this to work I had to recompile the parse-mspowerpoint plugin. Unfortunately there were some compilation errors there too that needed resolution.

file: nutch-0.7.1/src/plugin/parse-mspowerpoint/src/java/org/apache/nutch/parse/mspowerpoint/PPTConstants.java

```
  public static final String MIME_TYPE = "application/vnd.ms-powerpoint";
```

file: nutch-0.7.1/src/plugin/parse-mspowerpoint/src/java/org/apache/nutch/parse/mspowerpoint/MSPowerPointParser.java

```
// final PPT2Text ppt2text = new PPT2Text(input);
   final PPTExtractor ppt2text = new PPTExtractor(input);
```

With these two changes the MS PPT plugin would recompile and my PPT file fetching and parsing started working again.

Fetcher missing some URLs

So now I'm crawling and fecthing, parsing, indexing, etc. but I notice that some of the URLs that I want aren't being selected. So I hacked in a little bit of code to dump the data that the fetcher was retrieving and noticed that the HTML wasn't formatted correctly... how to fix that? I didn't have control over the pages that I was fecthing. So I stumble across the xml property parser.html.impl and the value **tagsoup**.

The default for the Nutch parser implementation property is **neko**. As it turns out this parser is pretty strict when it comes to enforcing HTML syntax. **tagsou** is exactly that... it finds pretty much anything that might be there. It works for me so it's what I use. I am thankful and impressed that both are available.

MS Word fetching fails

So again after the recompile and examining the crawl logs I notice that now my MS word documentation fetches are failing. There's a problem with the plugin that a little tweak would resolve. Note that this actually removes a bit of the metadata, but it's not significent to me so I don't mind the hack.

file: nutch-0.7.1/src/plugin/parse-msword/src/java/org/apache/nutch/parse/msword/WordExtractor.java

```
      // long editTime = si.getEditTime();
      long editTime = 0;
```

So by this point I've tweaked a few things to get the compile going and all of the fetching, parsing, indexing, etc. working simply using the crawl tool. To date, my crawl is very small, only about four thousand or so items, but keep in mind that most of that is actually documents that have to be parsed and many of them are megabytes in size themselves. It takes about a hour to crawl my limited intranet using a little PC.

Next for me includes adding in the MS excel plugin. I havn't been using it simply because the spreadsheets that we use are of less significence than the documents themselves and I've read a few comments to the effect that the excel plugin is working but not fully functional or correct 100% of the time.

Another interesting item might be to put some kind of feedback loop regarding system resources into the crawler. I find it very convienent to just use the crawl tool instead of having to write scripts to do everything. I would be nice to be able to have the crawl tool factor in memory such that should the tool begin to come close to some predefined ceiling it could dump a segment and begin another. Easier said than done I'm sure, but it would be handy.

I plan on scaling up my crawl to includes a few different locations on my intranet as I become familiar with the tools that are availalbe. At this point I'm ignorant of the full complement of features available to me, but then again my task is not as large as that of some of you out there with clusters indexing millions upon millions of pages. Nonetheless, my appreciation of what Nutch has given me compelled me to spend some time writing this up. Thanks Nutch team and I hope this helps some of you out there.

Regards, Marc