HowToContribute

Note: this page is

This page is a quick HOWTO explaining how to easily contribute patches to Nutch. It assumes you are using some form of Unix.

Getting the source code

First of all, you need the Nutch source code.

Create a directory in which you want to store the Nutch source code on your local drive, clone the Nutch git repository and cd into the Nutch project folder:

```
> cd somewhereOnYourDisk
> git clone https://github.com/apache/nutch.git
> cd nutch
```

Alternatively, you can use Apache's gitbox mirror. Please see UsingGit for further details.

Working time

Now it is time to work.

Feel free to modify the source code and add some (very) nice features using your favorite IDE.

But take care about the following points

- All public classes and methods should have informative javadoc.
- Unit tests are encouraged (http://www.junit.org).

Building a patch

First of all, please perform some minimal non-regression tests by:

- rebuilding the whole Nutch code
- executing the whole unit tests.

Building Nutch

```
> cd somewhereOnYourDisk/nutch
> ant
```

After a while, if you see

```
BUILD SUCCESSFUL
```

all is ok, but if you see

```
BUILD FAILED
```

please, read carefully the errors messages and check your code.

Unit Tests

```
> cd somewhereOnYourDisk/nutch
> ant test
```

After a while, if you see

```
BUILD SUCCESSFUL
```

all is ok, but if you see

```
BUILD FAILED
```

please, read carefully the errors messages and check your code. Detailed error logs are found in `build/test` (core classes) or `build/my-plugin/test` for plugins (here the plugin "my-plugin").

It is possible to run individual unit tests (useful during development):

- run unit tests from a single core test class (use class name without package path):

  ```
  ant test-core -Dtestcase=TestMimeUtil
  ```

- run unit tests for a specific plugin:

  ```
  ant test-plugin -Dplugin=protocol-okhttp
  ```

- or exclude test files by patterns:

  ```
  ant -Dtest.exclude='TestCrawlDb*.java **/TestNutchServer*' test
  ```

See also bin/nutch junit and WritingPluginExample-1.2#Unit_testing.

Functional Tests

If you are *perfectionist* you can also perform some functional tests by running Nutch. Please refer to the NutchTutorial

Opening a Pull-Request on Github

See the README on https://github.com/apache/nutch and the pull-request template how to open a pull-request on Github.

Creating a patch

Although a pull-request on Github is the prefered way of contribution, we still accept patches (not all contributors are on Github). In order to create a patch, just type from the root of the Nutch directory :

```
git diff --no-prefix > myBeautifulPatch.patch
vi myBeautifulPatch.patch
```

This will report all modifications done on Nutch sources on your local disk and save them into the *myBeautifulPath.patch* file. Then edit the patch file in order to check that it includes ONLY the modifications you want to add to the Nutch git repository.

Remember to generate a patch against a live branch, i.e. master for Nutch 1.x and 2.x for Nutch 2.x. The other branches are snapshots of past releases and the code might have evolved since then.

Proposing your work

Finally, patches can either be attached to a message sent to nutch-dev mailing list or to a bug report in Jira (*my prefered way in order to easily keep trace of contributions. But it is a very personal point of view*).

Testing and reviewing patches

Patches need careful testing and review to avoid regressions. Reviewing patches is mainly the task of committers. But you are welcome to help, esp. if you run into the same problem and found an issue in Jira, yet unresolved but with a patch attached. Review and testing needs time and should include the following steps:

1. try to **reproduce** the problem. If you're not able to reproduce a problem, it's impossible to test whether a patch really resolves it.
2. try to get a clear understanding of the problem
3. have a look at the patch file: Ideally, you'll get and understanding about the solution proposed by the patch (no problem if not)
4. apply the patch, see below #Applying_patches
5. build Nutch and test whether the problem disappeared
6. does the patch break other things (add regressions)?

- run the unit tests
- test situations which you feel they may be affected by the patch

1. report your finding in Jira or on nutch-dev. It's always better to have one review more, than to introduce a regression because of insufficient testing.

Applying patches

A properly generated patch can be automatically applied to the source tree. The `patch` utility is one tool to apply patches. Change into the Nutch root folder and run:

```
patch -p0 <path_to.patch
```

or

```
git apply path_to.patch
```

Do not ignore the output of `patch`, it may indicate errors. Applying a patch may (partially) fail, if the source code has changed meanwhile. A good starting point to learn more about patches it the Wikipedia article Patch.