

# README for Joycrawler

A spider program for hadoop applications



## Introduction

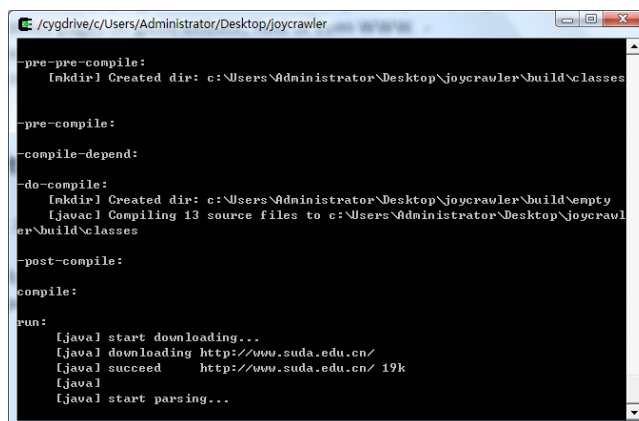
The project Joycrawler is a spider program for hadoop applications. Joycrawler itself is a standard hadoop program. Like the most popular spiders, Joycrawler downloads the internet pages on given website, host or even WWW.

Before you modify or develop on Joycrawler, you may want to run it in action and understand how it works.

## Getting Started

Prerequisites: JRE 6, Cygwin if Windows, Linux not required.

1. Copy the folder of Joycrawler to any place.
2. Simply type the command “ant run” under this folder, you might see this window below:



```
/cygdrive/c/Users/Administrator/Desktop/joycrawler
pre-compile:
[mkdir] Created dir: c:\Users\Administrator\Desktop\joycrawler\build\classes

pre-compile:

compile-depend:

do-compile:
[mkdir] Created dir: c:\Users\Administrator\Desktop\joycrawler\build\empty
[javac] Compiling 13 source files to c:\Users\Administrator\Desktop\joycrawler\build\classes

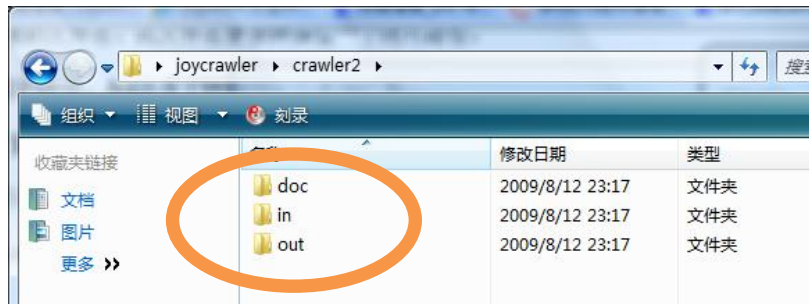
post-compile:

compile:

run:
[java] start downloading...
[java] downloading http://www.suda.edu.cn/
[java] succeed http://www.suda.edu.cn/ 19k
[java]
[java] start parsing...
```

After fetching all the pages or we reached the 15<sup>th</sup> iteration of fetching, the program will exit.

3. You can check the crawler folder under work directory to see your download results as following:



“Doc” folder is your webpage repository, and “out” folder contains all the link structure of the website you crawl.

## Reading the result from hadoop program

The folder doc and out are standard hadoop output folder, so you can use them directly in your hadoop program.

Both doc and out folders are filled with binary data, so you may want to use the SequenceFileInputFormat to read them in your hadoop program.

The Doc folder’s data pair is <Text, DocumentWritable>.

The Out folder’s data pair is <Text, OutlinksWritable>.

Here is a job configuration example of using “links” folder (replace it with your real “out” folder path) to do some link analysis:

```
public int run(String[] argo) throws Exception {
    // config a job and start it
    Configuration conf = getConf();
    Job job = new Job(conf, "Rank");
    job.setJarByClass(NormalizeDriver.class);
    job.setMapperClass(NormalizeMapper.class);
    job.setReducerClass(NormalizeReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setInputFormatClass(SequenceFileInputFormat.class);

    Path inPath = new Path("links");

    for (FileStatus s : FileSystem.get(conf).listStatus(inPath)) {
        Path sub = s.getPath();
        FileInputFormat.addInputPath(job, sub);
    }
}
```