Welcome to Apache Droids

Table of contents

1 What is this?

Droids aims to be an intelligent standalone robot framework that allows to create and extend existing droids (robots). In the future it will offer an administration application to manage and controll the different droids.

Droids makes it very easy to extend existing robots or write a new one from scratch, which can automatically seek out relevant online information based on the user's specifications.

Droids (plural) is not designed for a special usecase, it is a framework: **Take what you need, do what you want, impossible is nothing.**

It is the cocoon/UNIX philosophy for automated task processing in java. As a reminder a pipe in unix starts with an invoking component (which produces a stream) and then chain as much other components that interact on the stream that are needed. The modification of each component will be passed to the next component in the chain.

For example the following command in a unix box will lance a subversion command to check for the status on the local svn checkout (svn st). The next command will filter the files that are not under svn control (grep ?). The next command will modify the stream to create a command to add this files to the repository (awk ...). The last step will cause the invocation of the command by sending it to the shell (sh).

```
svn st | grep ? | awk '{print "svn add "$2}' | sh
```

In droids your are piping/processing your tasks with small specialist components that combined are resolving your task.

Droids offers you following the components so far:

- Queue, a queue is the data structure where the different tasks are waiting for service.
- Protocol, the protocol interface is a wrapper to hide the underlying implementation of the communication at protocol level.
- Parser -> Apache Tika, the parser component is just a wrapper for tika since it offers everything we need. No need to duplicate the effort. The Paser component parses different input types to SAX events.
- Handler, a handler is a component that uses the original stream and/or the parse (ContentHandler coming from Tika) and the url to invoke arbitrary business logic on the objects. Unless like the other components different handler can be applied on the stream/parse

A Droid (singular) however is all about **ONE special usecase**. For example the helloCrawler is a wget style crawler. Meaning you go to a page extract the links

and save the page afterward to the file system. The focus of the helloCrawler is this special usecase and to solve it hello uses different components.

In the future there could evolve different subprojects that are providing specialist components for a special use case. However if components get used in different usecases they should be considered common.

2 Feature list

- **Customizable.** Completely controlled by its default.properties which can be easily be overridden by creating a file build.properties and overriding the default properties that are needed.
- **Spring based.** The properties mentioned above get picked up by the build process which inject them in the spring configuration.
- **Extensible.** The spring configuration makes usage of the cocoon-configurator and its dynamic registry support (making extending droids a pleasure).
- **Multi-threaded.** The architecture is that a robot (e.g. HelloCrawler controls various worker (threads) that are doing the actual work.
- **Honor robots.txt.** By default droids honors the robot.txt. However you can turn on the hostile mode of a droid (droids.protocol.http.force=true).
- **Crawl throttling.** You can configure the amount of concurrent threads that a droid can distribute to their workers (droids.maxThreads=5) and the delay time between the requests (droids.delay.request=500). You can use one of the different delay components:
  - SimpleDelayTimer
  - RandomDelayTimer
  - GaussianRandomDelayTime

3 Architecture

The following graph shows the basic architecture of droids with the help of the first implementation (helloCrawler).

4 Why was it created?

Mainly because of personal curiosity and an usecase: The background of this work is that Cocoon trunk does not provide a crawler anymore and Forrest is based on it, meaning we cannot update anymore till we found a crawler replacement. Getting more involved in Solr and Nutch we saw request for a generic standalone crawler.

5 Requirements

Ant Optional Tasks

Important is that you have as well the optional Ant tasks installed! Otherwise you will not be able to build!

- Apache Ant version 1.7.0 or higher
- JDK 1.5 or higher

HEADSUP

!!! Please ONLY crawl localhost NEVER a internet site when you test the first time!!! You will need to adjust the urlfilters to limit loops.

6 Links / related projects

- Nutch web-search software
- The Web Robots Pages
- Programming webcrawler
- Writing a Web Crawler in the Java Programming Language
- Norbert
- Crawling AJAX
- Crowbar is a web scraping environment based on the use of a server-side headless mozilla-based browser.