Curation tasks in Jython

As mentioned in the "Scripted Tasks" chapter of Curation Tasks, you can write your curation tasks in several languages, including Jython (a flavour of Python running on JVM).
Instructions are outdated and unproven in DSpace 7.x

Setting up scripted tasks in Jython

1. Download the latest Jython installer jar (e.g. jython-installer-2.7.1.jar) from http://www.jython.org/downloads.html
2. Get `jython.jar` and the `Lib` directory.
   a. either unzip the installer jar:
      `unzip -d [dspace]/lib/ jython-installer-2.7.1.jar jython.jar Lib/`
      `unzip -d [dspace]/webapps/server/WEB-INF/lib/ jython-installer-2.7.1.jar jython.jar Lib/`
   b. or use it to install Jython:
      `java -jar jython-installer-2.7.1.jar --console`

      Note: Installation location doesn't matter, this is not necessary for DSpace. You can safely delete it after you retrieve `jython.jar` and `Lib`.
3. Install Jython to DSpace classpaths (step 2a already did this for you):
   a. The goal is to put `jython.jar` and the jython `Lib/` directory into **every** DSpace classpath you intend to use, so it must be installed in **both** `[dspace]/lib` and the webapp that deploys to Tomcat (if you want to run from the UI) - `[dspace]/webapps/server/WEB-INF/lib/`. There are no special maven/pom extensions - just copy in the jar and `Lib/`.
   b. You **can** use symlinks if you wish as long as allowLinking (Tomcat <=7, Tomcat 8) is set to true in that context's configuration. However, be warned that Tomcat documentation lists allowLinking="true" as a possible security concern.
   c. Note: Older versions of Jython mention the need for jython-engine.jar to implement JSR-223. Don't worry about that, new Jython versions, e.g. 2.7.1 don't require this.
4. Configure the curation framework to be aware of your new task(s):
   a. set up the location of scripted tasks in the curation system. This means simply adding a property to `[dspace]/config/modules/curate.cfg`:
      `script.dir=${dspace.dir}/ctscripts`
   b. in this directory, create a text file named "`task.catalog`". This is a Java properties file where lines beginning with '#' are comments. Add a line for each task you write. The syntax is following:

      ```
      # logical task name = script engine name|file name|constructor invocation
      mytask=python|mytask.py|MyTask()
      ```

      Notes:

      - **don't** put spaces around the pipe character or you'll get an error similar to this one:
        `ERROR org.dspace.curate.TaskResolver @ Script engine: 'python ' is not installed`
      - The "script engine name" is whatever name (or alias) jython registers in the JVM. You can use both "python" and "jython" as engine name (tested on jython 2.7.1).
      - The logical task name can't conflict with existing (java) task names, but otherwise any single-word token can be used.
      - The file name is just the script file name in the `script.dir` directory
      - "constructor invocation" is the language specific way to create an object that implements the task interface - it's ClassName() for Python
   c. If you want pretty names in the UI, configure other curate.cfg properties - see "`ui.tasknames`" (or groups etc)
5. Write your task.
   In the directory configured above, create your task (with the name configured in "`task.catalog`").
   The basic requirement of any scripted task is that it implements the ScriptedTask Java interface.
   So for our example, the `mytask.py` file might look like this:

   ```
   from org.dspace.curate import ScriptedTask

   class MyTask(ScriptedTask):
           def init(self, curator, taskName):
                   print "initializing with Jython"

           def performDso(self, dso):
                   print "perform on dso"
                   return 0

           def performId(self, context, id):
                   print "perform on id %s" % (id)
                   return 0
   ```

6. Invoke the task.
   You can do this the same way you would invoke any task (from command line, in the admin UI, etc). The advantage of scripting is that you do not need to restart your servlet container to test changes; each task's source code is reloaded when you launch the task, so you can just put the updated script in place.
   Example of invocation from command line:

   ```
   [dspace]/bin/dspace curate -t mytask -i 123456789/123 -r -
   ```

   Note: "`-r -`" means that the script's standard output will be directed to the console. You can read more details in the "On the command line" chapter of the Curation Tasks page.

See also

- Curation Tasks page in the official documentation
- Nailgun - for speeding up repeated runs of a dspace command from the command line
  Note: since DSpace 4.0, there's a solution for running dspace CLI commands in batch: Executing streams of commands
- Jython webapp for DSpace - general purpose (not curation task) webapp written in Jython, optionally with access to DSpace API