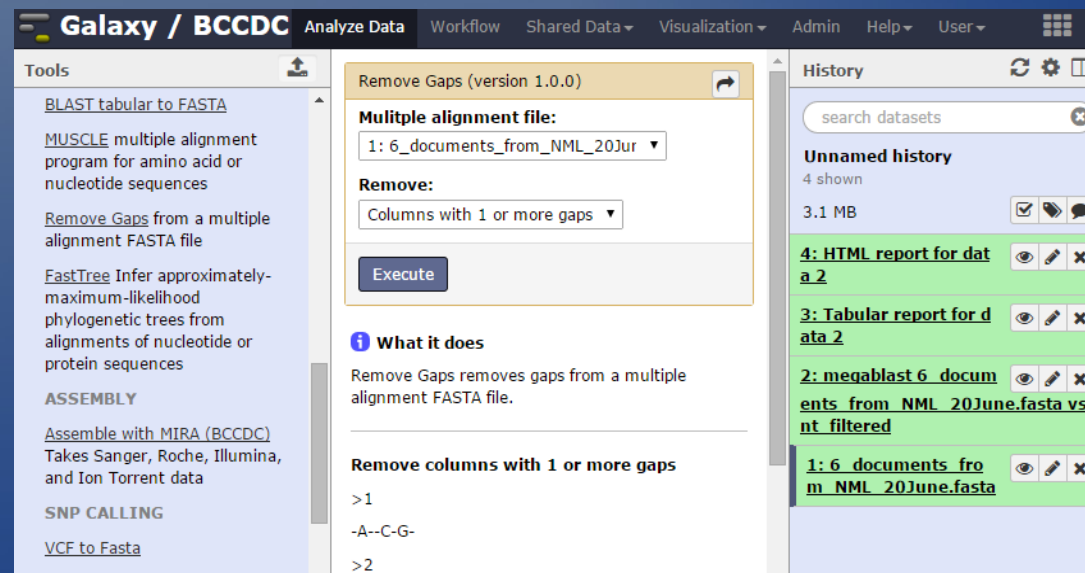# Write Your Own Galaxy Tool Wrapper

Build and test a Galaxy tool wrapper for your command line app!

You will create a form that a user can fill in and submit.

Results are placed into a user's history as a "dataset" that can be viewed, downloaded or used as part of a larger workflow.

Tool Development Process

Stages

- Develop command line script

- Craft XML tool definition file

- View & run via Galaxy

- Make and pass tests

- Add documentation

- Share

```xml
<tool id="cat1" name="Concatenate datasets">
    <description>tail-to-head</description>
    <command interpreter="python">
        catWrapper.py
        $out_file1
        $input1
        #for $q in $queries
            ${q.input2}
        #end for
    </command>
    <inputs>
        <param name="input1" type="data" label="Concatenate Dataset"/>
        <repeat name="queries" title="Dataset">
            <param name="input2" type="data" label="Select" />
        </repeat>
    </inputs>
    <outputs>
        <data name="out_file1" format="input" metadata_source="input1"/>
    </outputs>
    <tests>
        <test>
            <param name="input1" value="1.bed"/>
            <param name="input2" value="2.bed"/>
            <output name="out_file1" file="cat_wrapper_out1.bed"/>
        </test>
    </tests>
    <help>

.. class:: warningmark

**WARNING:** Be careful not to concatenate datasets of different kinds
(e.g., sequences with intervals). This tool does not check if the
datasets being concatenated are in the same format.

-----

**What it does**

Concatenates datasets
```

Anatomy of a tool

- A single tool definition file controls form display, field validation and app execution. XML tags specify the command to be run, fields for selecting input file(s) and other form parameters.

- Examples are in the tools/ folder of a Galaxy installation.

Galaxy Datasets

- A user's history tab contains "dataset" items. They are tool inputs and outputs.

- Galaxy has a data type system for specifying structured data formats (fasta, sam, bam, etc.), including ones that involve several files.

- Galaxy will automatically guess (sniff) the type of file a user has uploaded.

- You can control what (standard) type of input data your tool takes.

1: 6 documents fr
om NML 20June.fasta

6 sequences
format: **fasta**, database: **?**

uploaded fasta file

View details STABILIS_consensus_sequer

CCGGCGGCGATGGCCCGCCAGTGGGTCGATCTCGGCGG

CAAGCCGAGGAATCTCGAGGCGATCGAGGCGATCCTCC

TCCGCAGCCTCGAGACGATCGAGAAGTACCTCGACGCC

CCGGGCTTCCTGCAGGACGCGTGCACCGCGTTCGCGGC

GACCGACGGCTGGAGCAAGCTGACGGGCCACGAAGTGA

<command interpreter="...">
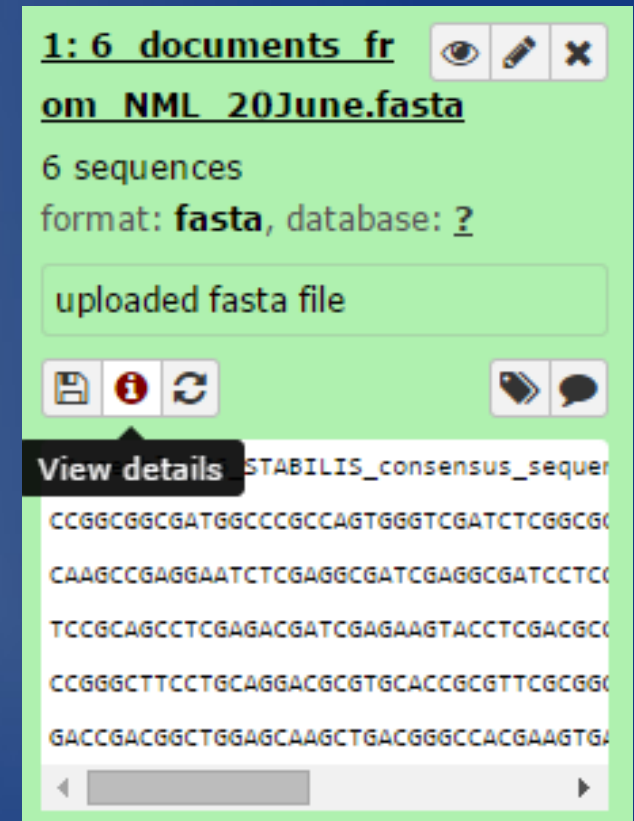
- The command is any linux shell command, or python, java, or other executable (that Galaxy knows of).

- It runs with the Galaxy user's permissions.

- Galaxy sets the tool's exact input and output file paths.

- Command tag allows Cheetah expressions.

```
<command interpreter="python">
    catWrapper.py
    $out_file1
    $input1
    #for $q in $queries
        ${q.input2}
    #end for
</command>
```

```
<command><![CDATA[
    cat "$input1" #for $q in $queries# ${q.input2} #end for# > "$out_file1"
]]></command>
```

```
Job              python /usr/local/galaxy/production1/galaxy-dist/tools/filters/catWrapper.py
Command-         /galaxy_data/production1/files/003/dataset_3977.dat
Line:            /galaxy_data/production1/files/003/dataset_3812.dat
                 /galaxy_data/production1/files/003/dataset_3821.dat
```

<inputs> <param type="...">

Each form field entry can be passed as a string value to a command line call parameter.

- "integer" : This shows some range validation on the input.

```
<param name="max_hits" type="integer" value="0" label="Maximum
hits to show" help="Use zero for default limits">
    <validator type="in_range" min="0" />
</param>
```

- "float" : Input accepts decimal numbers

```
<param name="evalue_cutoff" type="float" size="15" value="0.001"
label="Set expectation value cutoff" />
```

- "boolean" :

```
<param name="ungapped" type="boolean" label="Perform ungapped alignment only?"
truevalue="-ungapped" falsevalue="" checked="false" />
```

<inputs> <param type="data">

- type="data" : provides a file selector of one or more input files from current history.

```
<inputs>
    <param name="input1" type="data" label="Concatenate Dataset"/>
```

- You can constrain this to files of a particular data type.

```
<param name="query" type="data" format="fasta"
label="Nucleotide query sequence(s)"/>
```

- Add ' multiple="true" ' : allows selection of more than one file. These will be passed as a comma-delimited list.

- Or use a <repeat> to collect associated parameter inputs.

```
<repeat name="queries" title="Dataset">
    <param name="input2" type="data" label="Select" />
</repeat>
```

<inputs> <param type="...">

- "select" : drop down list or radio/checkboxes

```
<param name="blast_type" type="select" display="radio" label="Type of BLAST">
    <option value="megablast">megablast</option>
    <option value="blastn">blastn</option>
    <option value="blastn-short">blastn-short</option>
    <option value="dc-megablast">dc-megablast</option>
</param>
```

- "text" : provides text input field.

- "data_table" : Filter and display rows / columns from a Galaxy-registered tab-delimited table

  ... and others. See:

https://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

<outputs><data name="...">

The <outputs> section has one or more <data ...> tag output file specifications. For each data tag Galaxy will set up a new dataset file path to output your app's data to.

```
<outputs>
    <data name="output1" format="tabular"
label="${blast_type.value_label} on ${db_opts.db_opts_selector}">
    </data>
</outputs>
```

format="..." : Contains one of Galaxy's data types (see admin tab > Administrator > Server > View data types registry).

The "label" shows up as the dataset name of a history record.

If the number of output files varies, Galaxy doesn't list each one in history. One's tool must output an HTML report to show links to each of them.

Error handling

Apps generate exit code=0 for normal operation; higher numbers are error or warning codes. How Galaxy should interpret them is up to you; normally 0 means "green" ok; anything else triggers a red "failed" state.

The <stdio> tag holds any codes you want to upgrade or downgrade the error state for.

Robot from Lost in Space (1965)

```
<stdio><exit_code range="1:" level="fatal"/></stdio>

<stdio>
     <exit_code range="2"   level="fatal"   description="Out of Memory" />
     <exit_code range="3:5" level="warning" description="Low disk space" />
     <exit_code range="6:"  level="fatal"   description="Bad input dataset" />
     <!-- If the return code has not been set propery check stderr too -->
     <regex match="Error:" />
     <regex match="Exception:" />
</stdio>
```

Tests

- Not required for local use, but for a shared tool, tests are expected.

- Specify input parameters by name and value as they would be entered in form.

- The input and output files are specified in tool's "test-data" folder.

```
<test>
    <param name="input1" value="1.bed"/>
    <param name="input2" value="2.bed"/>
    <param name="input2" value="3.bed"/>
    <output name="out_file1" file="cat_wrapper_out2.bed"/>
</test>
```

- Normally time consuming to test in Galaxy directly, so we use "planemo"

Help Info

"Free" text can only be inserted at form bottom below the Execute button. It uses reStructuredText markdown format:
http://docutils.sourceforge.net/docs/user/rst/quickref.html

Add a references section if your app uses 3rd party software or if it is based on your own paper. This can simply be a help section **References** header and text. It can also include an automated lookup:
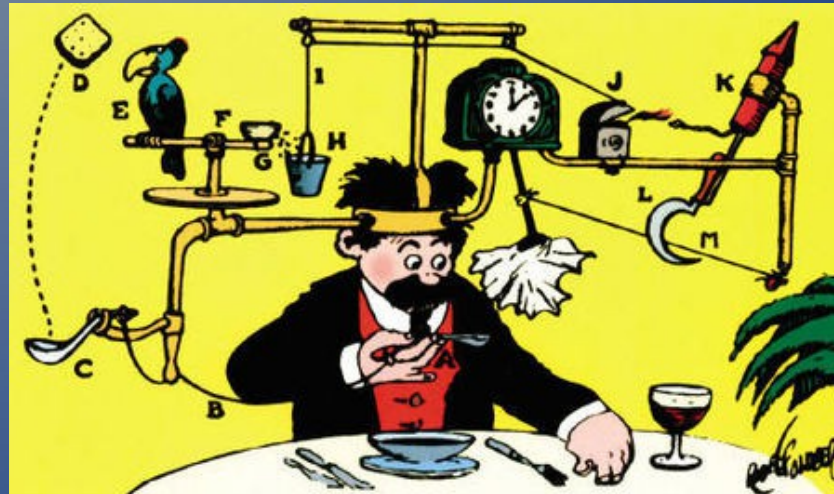
```
<help><![CDATA[

.. class:: warningmark

**WARNING:** Be careful not to concatenate
datasets of different kinds (e.g., sequences with
intervals). This tool does not check if the
datasets being concatenated are in the same
format.

-----

**What it does**

Concatenates datasets
]]></help>
```

```
<citations>
    <citation type="doi">10.7717/peerj.167</citation>
</citations>
```

Ready?

Self-Operating Napkin
(Copyright © Rube Goldberg™
& © of Rube Goldberg, Inc.)

Editing Issues

Normally to load your new tool into Galaxy requires manually adding it to the Tool Menu and then restarting Galaxy!

Only A Galaxy **admin** can test changes in a tool's form XML by using "Reload Tool" as shown to right.

Editing your tool's main app code shouldn't require a Galaxy restart. Editing other modules (that your script imports) may require a restart (due to caching).
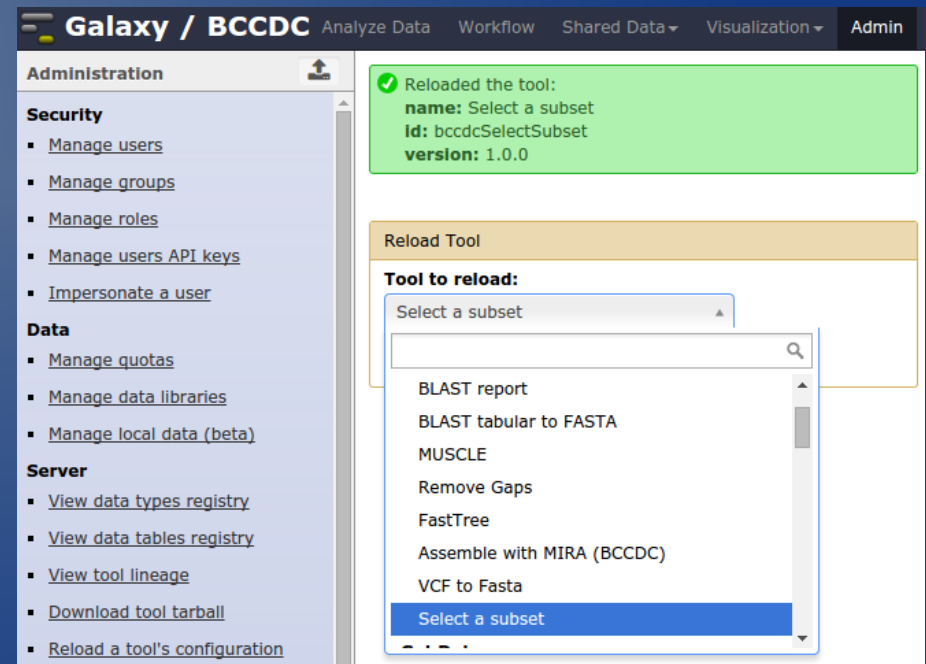
Editing Issues

And if tool actually loads, running tests takes forever ...

- Galaxy will crash if a tool's XML is broken (incomplete tags, bad nesting, bad characters). Galaxy won't load tool, or may not restart.

- Buried in web0.log or paster.log one will find reference to the tool not loading.

- XML character escaping: $: \$ & : &   < : <   etc.

- **OR** wrap <command> and <help> content in <![CDATA[...]]>

Get Started on your Galaxy Server

Solution: The "Planemo" app: It validates tool's XML, runs Galaxy tests, and launches tool in trimmed-down Galaxy. You don't have to be an admin to do this.

Use a terminal to ssh in with -X to enable X-windows apps.

Navigate to the Galaxy installation's tools/TESTING/ subfolder.

> cd /usr/local/galaxy/galaxytest/galaxy-dist/tools/TESTING/

Make a folder for your tool:

> mkdir randomish; cd randomish

Using Planemo

Try planemo out on a demo project template:

> planemo project_init –template=demo

> gedit randomlines.xml &   // Edit demo tool

> planemo lint   // Validate XML

> planemo test   // Run tool tests

> firefox *.html &   // View test results

> planemo view   // Run it in Galaxy (Ctrl-c cancels)

For a great intro to planemo's other features, see
https://planemo.readthedocs.org/en/latest/writing_standalone.html

Links

The Galaxy developer's email list is a great place to get help
http://dev.list.galaxyproject.org

Galaxy Documentation
https://wiki.galaxyproject.org/Admin/Tools/WritingTests
https://wiki.galaxyproject.org/Admin/Tools/AddToolTutorial
https://wiki.galaxyproject.org/Tools/BestPractices

The Galaxy tool parameter syntax, though its not complete
https://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax

Planemo Tool tester and Tutorial
http://planemo.readthedocs.org

Cheeta expression language
http://cheetahtemplate.org/learn.html