Writing Galaxy Tools

30 June 2014, GCC2014, Baltimore

The James Hutton Institute

Peter Cock[1], Björn Grüning[2], Greg Von Kuster[3]

[1] James Hutton Institute, Scotland, UK; [2] Albert-Ludwigs-University, Germany; [3] Penn State University, USA

Wrapping Command-line Tools

- Tell Galaxy what options to show the user
- Galaxy tells your tool the selected input filenames
- Galaxy tells your tool the desired output filenames
- Must tell Galaxy how to invoke the underlying tool...

Heart of each Galaxy tool is an XML file

Core elements:

- `<inputs>` – parameters/options/files
- `<outputs>` – output files expected
- `<command>` – how to turn this into a command line string

Secondary elements:

- `<requirements>` – tell Galaxy how to find the binaries etc
- `<stdio>` – what counts as an error?
- `<description>` – subtitle describing tool
- `<help>` – instructions to show the end user
- `<tests>` – functional tests

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

Heart of each Galaxy tool is an XML file

Example:

```
<tool id="my_tool" name="My Tool" version="0.0.1">
  <command>my_tool "$input1" "$output1"</command>
  <description>Run My Tool (patent pending)</description>
  <inputs>
    <param name="input1" type="data" format="fasta"
           label="Sequence in" help="FASTA format." />
  </inputs>
  <outputs>
    <data name="output1" format="fasta"
          label="My Tool Results" />
  </outputs>
  <help>
    This is a Galaxy wrapper for My Tool.
  </help>
</tool>
```

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

The `<inputs>` and `<param ...>` tags

- `<inputs>...</inputs>` contains `<param ...>` tag(s)
- Each `<param ...>` tag requires a unique `name`
  - This is used in the `<command>` and `<tests>`
- Each `<param ...>` tag requires a `type`
  - e.g. for `type="data"` for an input file
- Each `<param ...>` tag should have a `label` and `help`
  - These are shown in the user interface
- There are additional type-specific attributes
  - e.g. for `type="data"` add `format="..."` for file type

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

The `<inputs>` and `<param ...>` tags

Use `<param type="???" ...>` to control each parameter:

- `type="data"` – input file (from current history)
- `type="text"` – any string via a text box
- `type="integer"` – whole number via a text box
- `type="float"` – arbitrary number via a text box
- `type="select"` – Drop down lists, or radio buttons
- `type="boolean"` – True/false value with checkbox
- `type="data_column"` – Pick column(s) from a tabular file
- ...

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

The `<outputs>` and `<data ...>` tags

- `<outputs>...</outputs>` contains `<data ...>` tag(s)
- Each `<data ...>` tag requires a unique `name`
  - This is used in the `<command>` and `<tests>`
- Each `<data ...>` tag should have an `ftype`
  - e.g. for `ftype="fasta"` for a FASTA output file
- Each `<data ...>` tag should have a `label`
  - This is the default dataset description in the history

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

The `<command>` tag – basics

- The `<command>` tag is a command line string template
- Named every input `<param ...>` and output `<data ...>`
- Use `$name` to refer to that input parameter or output file
  - Ideally use `"$name"` in case of spaces in filename
- Can split `<command>` over multiple lines
- XML so need `&amp;`, `&lt;` and `&gt;`
- Must escape `$` to get an actual dollar sign, e.g.

```
<command>
my_tool --threads \$GALAXY_SLOTS
"$input1" "$output1"
</command>
```

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

The <command> tag – advanced

- The <command> tag uses the Cheetah template language
- This can include for loops and if statements, e.g.

```
<command>
my_tool --threads \$GALAXY_SLOTS
## double hash for comment lines
## single hash for Cheetah syntax
#if $output_choice=="long"
--long
#end if
"$input1" "$output1"
</command>
```

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

Advanced parameter options

- Galaxy supports conditional and repeated constructs
  - Defined with more XML in the `<inputs>` section
- Galaxy supports multiple input files as one parameter
- Requires Cheetah syntax in the `<command>` tag
- Easiest to learn by example?

```
$ cd galaxy-dist
$ grep "<conditional " tools/*/*.xml
...
$ grep "<repeat " tools/*/*.xml
...
```

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

The <help> tag

- Uses reStructuredText markup language
- Blank line for a paragraph break
- Use asterisks for *italics*, double-asterisks for **bold**
- Can include tables, images, links, etc.

```
<tool id="my_tool" name="My Tool" version="0.0.1">
  ...
  <help>
    This is a Galaxy wrapper for *My Tool*.

    If you use this tool, **please cite this paper**:
    ...
  </help>
</tool>
```

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

The <stdio> tag

- The <stdio> tag controls error detection
- Galaxy default is any output on **stderr** means an error (!)
- Unix/Linux convention allows logging etc on **stderr**
- Unix/Linux convention is non-zero return code means error

For Unix style tools, only check the return code:

```
<tool id="my_tool" name="My Tool" version="0.0.1">
  ...
  <stdio>
    <!-- Anything other than zero is an error -->
    <exit_code range="1:" />
    <exit_code range=":-1" />
  </stdio>
  ...
</tool>
```

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

Further reading

- Functional Tests
- Dynamics captions on output files
- Variable numbers of output files
- Composite datatypes
- Defining new Galaxy datatypes
- Dependencies & The Tool Shed
- Galaxy macros to reduce repetitive XML