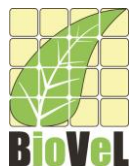




Tool service

Aleksandra Pawlik
myGrid Team
University of Manchester

VLIZ, 2014-10-06 / 2014-10-08
<http://www.taverna.org.uk/>



This work is licensed under a
[Creative Commons Attribution 3.0 Unported License](http://creativecommons.org/licenses/by/3.0/)





What is a tool service?

- Allows you to call a command line script as part of a workflow
 - Simplest case is calling a single tool
- Can be run on your local machine or a machine that you can ssh to
- Data is passed by reference
 - No big transfers to/from Taverna
- Data kept where the script is run until/unless needed



Using a simple tool service

- Choose “Tool” from the “Insert” menu
- In the tool service popup type
java -version
- Close the configuration
- Connect the STDERR and STDOUT ports of the tool service to workflow output ports



Simple tool service configuration

Workflow1:Tool

Command String replacements File inputs File outputs Advanced Location

Specify the commands that you want to run. You can use data arriving at an input port to replace parts of the command or to write to a file. You can also take data written to a file and send it to an output port.

```
java -version
```

Line: 1 Column: 13

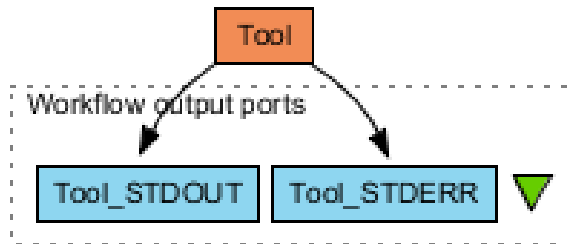
Valid return codes:

Show STDIN Show STDOUT Show STDERR



Simple tool workflow

- Run the workflow



- STDERR should look similar to:
`java version "1.8.0"`
`Java(TM) SE Runtime Environment (build 1.8.0-b132)`
`Java HotSpot(TM) 64-Bit Server VM (build 25.0-b70, mixed mode)`



Downloading an example tool

- We are going to use the *forester* utilities by [Christian Zmasek](#)
- Download
 - forester_1037.jar as by following the links on <https://sites.google.com/site/cmzmasek/home/software/forester/phyloxml-converter>
 - If you get a Google Drives doc rightclick and Save link as..
 - ..or download it from the myExperiment group
 - See <http://www.myexperiment.org/files/1316.html>
- Remember which folder you downloaded it to
 - **Your will have to change “C:\Users\stain\Downloads” to this folder**



Calling the example tool - 1

- Create a new workflow with a tool service that calls the jar (modify the path)

```
java -cp
```

```
C:\Users\stain\Downloads\forester_1037.jar
```

- Connect STDERR and STDOUT
- Run the workflow
- It fails. We cannot just call the jar



Calling the converter - 1

- We cannot just call the jar
- Look for the parameters of this tool at <https://sites.google.com/site/cmzmasek/home/software/forester/phyloxml-converter>
- Change the tool service so the script says on one line:

```
java -cp C:\Users\stain\Downloads\forester_1037.jar  
org.forester.application.phyloxml_converter -f=nn  
infile outfile
```
- This converts the **infile** to PhyloXML and writes it to **outfile**
- Run the workflow



Calling the converter - 2

- We need to pass an input file
- Configure the tool service and add a file input called infile

Workflow3:Tool

Command String replacements File inputs File outputs Advanced Location

You can use a file input to feed data into the service via an input port and have that data written to the specified file.

Taverna port name:

Use port name for file:

To file:

File type:

Remove

Add file input

Help Apply Close



Calling the converter - 3

- Add a file output called outfile

Workflow3:Tool

Command String replacements File inputs File outputs Advanced Location

You can use a file output to take the content of a file produced by the tool and send it to an output port of the service.

Taverna port name:

Use port name for file:

From file:

File type:

Remove

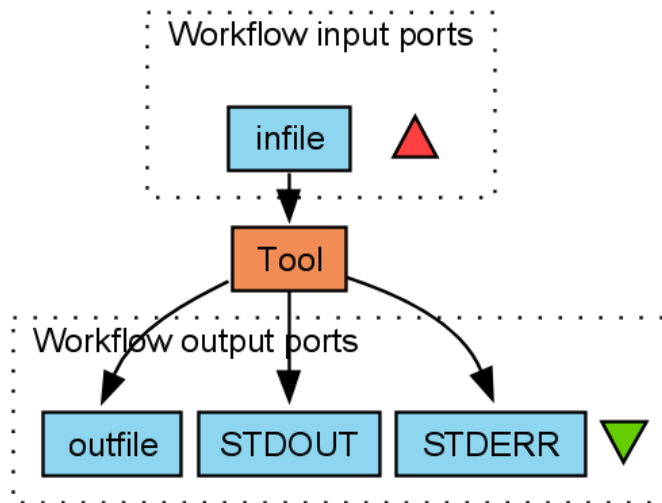
Add file output

Help Apply Close



Calling the converter - 4

- The tool service now has two extra ports
- Connect infile to a workflow input port and outfile to a workflow output port





Calling the converter - 6

- Run the workflow
- As input, you can use the contents of <http://www.myexperiment.org/files/1055/versions/1/download/example.nh.txt>

(or use **Set URL**)

- The *outfile* is in PhyloXML format
 - Click **Value type: XML tree**



Showing the PhyloXML - 1

- Rename the first tool to **converter**
- Add a new tool service that calls

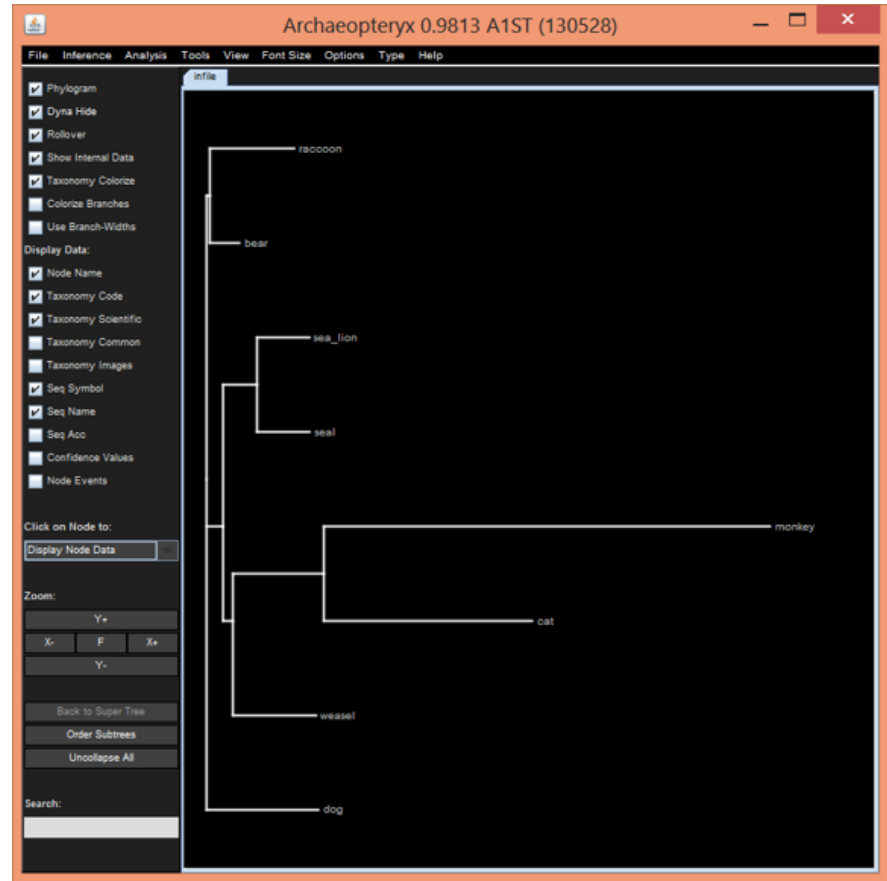
```
java -cp C:\Users\stain\Downloads\forester_1037.jar  
org.forester.archaeopteryx.Archaeopteryx infile
```

- Add a file input called **infile**
- Rename the tool service to **display**
- Connect the **outfile** of converter to the **infile** of **display**
- Run the workflow



Showing the PhyloXML - 2

- The archaeopteryx display tool will show
- Exit it to finish the run





Using string replacement - 1

- PhyloXML **converter** can take options
- Add a new String replacement port to the converter service called **options**

The screenshot shows a web-based interface for configuring a workflow converter. The window title is 'Workflow3:converter'. It has several tabs: 'Command', 'String replacements', 'File inputs', 'File outputs', 'Advanced', and 'Location'. The 'String replacements' tab is active. Below the tabs, there is a text box with the instruction: 'You can use a string replacement to feed data into the service via an input port and have that data replace part of the command.' Below this, there is a form with three fields: 'Taverna port name:' with the value 'options', 'Replace port name:' with a checked checkbox, and 'String to replace:' which is empty. A 'Remove' button is located to the right of the 'String to replace' field. At the bottom right of the form area, there is an 'Add string replacement' button. At the very bottom of the window, there are 'Help', 'Apply', and 'Close' buttons.



Using string replacement - 2

- Change the converter script to include the options
`java -cp C:\Users\stain\Downloads\forester_1035.jar org.forester.application.phyloxml_converter -f=nn %%options%% infile outfile`
 - `%%options%%` will be replaced by the string passed to the service
 - Connect the options port to a workflow input port
 - Run the workflow with options as the empty string
 - Run the workflow with options as `-o`
 - Compare the *outfile* with that from the previous run



Further exercises

- Add the **Xpath service** to pick up the species name of the second-level clade branch (bear, raccoon)
- Create a **component** family in your local registry called *forester*
- Create a components in the forester family for the *converter* and *display* services
- Build a workflow using the two components from *Available Services*
- What possible problems can you imagine if you want to share a workflow using the External Tool service?
- Expert: Are you able to modify your workflow to be sharable? Hint: Look at **Advanced** tab of Tool service.