Extension of Galaxy to Utilize Web Services and A Semantic Suggestion Engine

Alok Dhamanaskar, Akshay Choche, Michael E. Cotterell, Jie Zheng, Christian Stoeckert Jr., Jessica C. Kissinger and John A. Miller
University of Georgia and University of Pennsylvania, USA
The Motivation:

Galaxy is currently restricted to tools/data resources that are wrapped specifically for Galaxy and manually added either via and tool wrapper and addition to tools_conf.xml or addition via the tool shed.

Wouldn’t it be nice if Galaxy could utilize any tool that is publicly exposed via a Web service with just a click?
What are Web services?

• A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. Because all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

• Web services come in two flavors:
  – SOAP services described by WSDL XML documents
  – RESTful services described by WADL XML documents
Enabling Galaxy to Invoke Web services

**Prerequisites:**
- SSH/Terminal Access to Galaxy Installation
- Java 7 SE or EE (Oracle or OpenJDK)
- Python 2.6+
- JPype Python Library
Enabling Galaxy to Invoke Web services

Installation is Easy!

1. Set Environment Variables

```
galaxy@galaxy:~/galaxy-dist$ export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
galaxy@galaxy:~/galaxy-dist$ export GALAXY_HOME=$(pwd)
galaxy@galaxy:~/galaxy-dist$
```
Enabling Galaxy to Invoke Web services

Installation is Easy!

1. Set Environment Variables
2. Download Installation Script

```
galaxy@galaxy:~/galaxy-dist$ cd
100%

2013-06-21 12:06:17 (111 MB/s) - `WSExtensionTool.zip' saved [41656528/41656528]
galaxy@galaxy:~/galaxy-dist$ unzip WSExtensionTool.zip
```
Enabling Galaxy to Invoke Web services

Installation is Easy!

1. Set Environment Variables
2. Download Installation Script
3. Run Installation Script

```
galaxy@galaxy:~$ cd WS\ Extension\ Tool/
galaxy@galaxy:~/WS Extension Tool$ python install.py
Preparing to install the new WS Extension tool
Copied the folder 'WebServiceToolWorkflow_REST_SOAP' to /home/galaxy/ga
The "Add Web Service Tool" added successfully
The "Web Service Tools" added successfully
The section "Web Service Workflow Tools" is added successfully
Workflow disabling Code added to __init__.py
Refresh Code added to __init__.py
Refresh code is added to app.py
****Installation of tool is complete. Now reload galaxy to use the inst
galaxy@galaxy:~/WS Extension Tool$
```
Enabling Galaxy to Invoke Web services

Installation is Easy!

1. Set Environment Variables
2. Download Installation Script
3. Run Installation Script

Now the Web Service tool has been installed and added to Galaxy’s tool_conf.xml file.

WEB SERVICE TOOLS

Add Web service tool(s)
- Step a: Enter the URL (WSDL/WADL location) of tool (web service) description document
- Step b: Select desired Web Service Tools / Operations to be added.
- Step c: Add tool(s) to Galaxy

However, you need to Restart Galaxy for tool_conf.xml changes to take effect!
Enabling Galaxy to Invoke Web services

Installation is Easy!

1. Set Environment Variables
2. Download Installation Script
3. Run Installation Script

However, you need to

Restart Galaxy

for tool_conf.xml changes to take effect!
tool_conf.xml

Workflow Client XMLs

Web Service Python Script

Web Service Invocation Library

JPype

Web Service

Galaxy Architecture with Web Services
Using Web Services in Galaxy

WEB SERVICE TOOLS

Add Web service tool(s)
- Step a: Enter the URL (WSDL/WADL location) of tool (web service) description document
- Step b: Select desired Web Service Tools / Operations to be added.
- Step c: Add tool(s) to Galaxy

Step a: Enter the URL (WSDL/WADL location) (version 1.1.0)

Enter the url of the tool's description document:
http://mango.ctegd.uga.edu/jkissingLab/SWS/Wsanno

Execute

The following job has been successfully added to the queue:

30: Step a: Enter the URL (WSDL/WADL location)

You can check the status of queued jobs and view the resulting data by refreshing the History pane. When the job has been run the status will change from 'running' to 'finished' if completed successfully or 'error' if problems were encountered.
Using Web Services in Galaxy

**WEB SERVICE TOOLS**

**Add Web service tool(s)**
- Step a: Enter the URL (WSDL/WADL location) of tool (web service) description document
- Step b: Select desired Web Service Tools / Operations to be added.
- Step c: Add tool(s) to Galaxy

**Tool Configuration**

- tool_conf.xml

**Workflow Client XMLs**

```
1. The previous step was (No action required):
   30: Step a: Enter the URL
You can also select any previously performed Step a from the history to avoid repeated execution of Step a.

2. The tool specified by the Web service is:
   http://mango.ctegd.uga.edu/jkissingLab/SWS/Wsannotation/resources/clustalw
   Displays the tool (Web service) chosen in Step a. No action required.

3. Select functions of the Web service tool to add to galaxy:
   Select All  Unselect All
   - run
   - getStatus
   - getResultTypes
   - getResult
   - getParameters
   - getParameterDetails
   Some tools have more than one function. It is important to select the appropriate functions.
```

Generates Workflow Client XML Files
Using Web Services in Galaxy

WEB SERVICE TOOLS

Add Web service tool(s)
- Step a: Enter the URL (WSDL/WADL location) of tool (Web service) description document
- Step b: Select desired Web Service Tools / Operations to be added.
- Step c: Add tool(s) to Galaxy

Step c: Add tool(s) to Galaxy (version 1.1.0)

Execute to:
- Add Tool

Does Nothing!

Need to

Restart Galaxy
for tool_conf.xml changes to take effect

tool_conf.xml

Workflow Client XMLs
Using Web Services in Galaxy

WEB SERVICE TOOLS

Add Web service tool(s)

* Step a: Enter the URL (WSDL/WADL location) of tool (web service) description document

* Step b: Select desired Web Service Tools / Operations to be added.

* Step c: Add tool(s) to Galaxy

Step c: Add tool(s) to Galaxy (version 1.1.0)

Precite to:

Add Tool

Nothing!

Need to Restart Galaxy for tool_conf.xml changes to take effect
Web Services can be used in Workflows
Find Services / Tools for My Workflow

Seems like a lot of services to look through! I wonder if I missed any?

Total Services = 483 with maybe 1000s of operations

Popular Web Service Registries

Tools available in Galaxy + Tools in the Galaxy Tool-shed
I selected 2 of those operations, entered them in my workflow designer, but when I tested, it didn’t work.
Connect the Newly Added Service into the Workflow

Wish I could figure out which service operation(s) might fill the gap in my workflow.

Why can’t the software make a suggestion?

Workflow design / service composition
Semantically Annotated Web Services

wublast.sawSDL – Semantically Annotated WSDL for WU-BLAST

http://purl.obolibrary.org/obo/OBIws_0000082

Ontology – OBIws

- standard error
- sequence similarity expectation value
- signal peptide prediction threshold value
- quantitative confidence value
- is-a
Modified the UI using minimal mako modifications and some jQuery magic!
Service Suggestion

1. Choose the type of suggestion you want

Forward Suggestion

Backward Suggestion

Bidirectional Suggestion

Choose Suggestion Type:
Choose the kind of suggestions you want by indicating where you think they should go in the workflow.

Click here for more information about the different types of service suggestion.

Tools to Feed From:
Select the tools in current workflow that you think should feed into suggested service operations.

Step 2 – wublast getResult

Goal / Purpose:
In simple terms, describe what you are trying to do/find (e.g., multiple
Service Suggestion

1. Choose the type of suggestion you want

2. Select Relevant Tools

The tools currently on the canvas will appear in one or more dropdown lists (depending on suggestion type).

Select specific ones to utilize for suggestion or choose “all”.
1. Choose the type of suggestion you want

2. Select relevant tools

Optionally, specify a goal
Service Suggestion

1. Choose the type of suggestion you want

2. Select relevant tools

Optionally, specify a goal

If you’re not sure what something means then click on “help” to display a modal help dialog.
Service Suggestion

1. Choose the type of suggestion you want
2. Select relevant tools
   Optionally, specify a goal
3/4. Run the query to get suggestions!

Once you’ve specified what you want, click on “Run Query” to get suggestions!
Annotated services are displayed in the results, ranked by how well they satisfy the suggestion query.
Clicking on a tool adds it to the canvas!
Why not annotate Galaxy Tools to enable the SSE?

**tabular_to_fasta.xml – Example of a Semantically Annotated Galaxy Tool**

```xml
<description>converts tabular file to FASTA format</description>
<command interpreter="python">tabular_to_fasta.py $input $title_col $seq_col $output</command>
<inputs>
  <param name="input" type="data" format="tabular" label="Tab-delimited file" />
  <param name="title_col" type="data_column" data_ref="input" multiple="True" numerical="False" label="Title column(s)" help="Multi-select list – hold the appropriate key while clicking to select multiple columns" />
  <param name="seq_col" type="data_column" data_ref="input" numerical="False" label="Sequence column" />
</inputs>
<outputs>
  <data name="output" format="fasta" modelReference="http://purl.obolibrary.org/obo/OBIws_0000106" />
</outputs>
```

All we need to do is add modelReferences to a tool’s XML file!

Ontology – OBIws
Annotating Services with RadiantWeb

Simplified View of XML File

Tree View of Ontology
Annotating Services with RadiantWeb

Radiant Web - Semantic Annotation Tool

WSDL/WADL Viewer

Ontology Viewer

- Identified Operations in the WSDL (portType: DispatcherService)
  - getParameters
  - getStatus
  - check web service execution status objective
  - Inputs (getStatusRequest)
  - Outputs (getStatusResponse)
  - status(*)
  - web service execution status
  - getResultTypes
  - getResult
  - run
  - getParameterDetails

Legend:
- Approve Suggested Term
- Reject Suggested Term
- Remove annotation
- Suggested Term
- Pre-existing annotation
- Approved Term
- Operation
- Messages (Input/Output)
- ComplexType (Inputs/Outputs)
- SimpleType (Inputs/Outputs)

Version 1.0 The project is developed at The University of Georgia. For feedback and suggestions contact us.
Annotating Services with RadiantWeb

Documentation

Inputs/Outputs

WSDL/WADL Viewer

WSDL location: C:\fakepath\wublast.wsdl

Browse Load WS

WSDL: C:\fakepath\wublast.wsdl

Save WSDL Recommend Terms WSDLXML View

GetParameters

getStatus

check web service execution status objective

Outputs (getStatusResponse)

status(*)

web service execution status

getResultSetTypes

getResult

generate

getParameterDetails

Ontology Viewer

Search:

Definition

Thing

- obsolete class
- entity
  - occurring
  - spatial region
  - dependent spatial region
- independent spatial region

Legend

- Approve Suggested Term
- Reject Suggested Term
- Remove annotation
- Suggested Term
- Pre-existing annotation
- Approved Term
- Operation Messages (Input/Output)
- ComplexType (Inputs/Outputs)
- SimpleType (Inputs/Outputs)

Version 1.0 The project is developed at The University of Georgia. For feedback and suggestions contact us.
Adding Web services to Galaxy

Advantages
• Increased tool/database access
• Don’t need to wrap the tools/data access
• Distribute the load - Utilize someone else’s computers
• Further empower the end user

Disadvantages
• Finding the Web services
• Moving data
• Hidden analysis step (Get job ID)
• All possible result formats (many not compatible with existing Galaxy tools)
• Security
• Banned from sites (depends on who is viewed as the user)
Adding a Semantic Suggestion Engine

Advantages

• Helps users find Galaxy tools with the desired functionality
• Helps with workflow construction
• Annotated tools are better documented tools. You will now what that parameter does and understand inputs and outputs

Disadvantages

• The community needs to annotate its tools
• Not all semantic terms will exist
• You give up some display area in the history column when the SSE is being used. It can be collapsed when not in use.
Thank you!

http://mango.ctegd.uga.edu/jkissingLab/SWS/