Developers Guide

This guide is intended for the developers who wish to further enhance the Web Service Extensions to Galaxy. This document provides an overview of the code and the flow of control within the code for adding web service operation as Tools to the Galaxy.

We assume that you have an instance of Galaxy running with our Web Service Extensions tool installed. For installation instructions please refer to the installation guide. We would also assume that you are familiar with using the tool, if you are not please refer to the users guide.

The **Add web service Tools** section that we have added comprises of three tools (to be executed sequentially as three steps) pointing to three XML tool description files each of these invoking python files for respective purpose.

- WebServiceTool_input_url_REST_SOAP.xml ->WebServiceTool_input_url_m.py
- WebServiceTool_input_method_REST_SOAP.xml ->WebServiceTool_input_method_m.py
- refreshTool.xml -> refreshTool.py

As that is the way to add tools to Galaxy; if you are not acquainted with this, please refer to https://bitbucket.org/galaxy/galaxy-central/wiki/AddToolTutorial

Step a
WebServiceTool_input_url_REST_SOAP.xml -> WebServiceTool_input_url_m.py

The XML file, describes the tool as having one input for taking the url of the description document. It invokes WebServiceTool_input_url_m.py for processing the description file.

WebServiceTool_input_url_m.py

The module checks if the description document is for a SOAP Web service or a REST Web service and invokes a method in getMethods.py accordingly.
→ Invokes getWSDLMethods() in case of SOAP web services.
→ Invokes getWADLMethods() in case of REST web services.
These functions have been defined in getMethods_m.py

getMethods_m.py

This module has 3 methods namely
- getWSDLMethods(),
- getWADLMethods() and
- getSAWADLMethods()

getWSDLMethods() :

→ The method makes use of ZSI and creates client stubs by invoking “wsdlurl2path” method in “wsdl2path.py” module in package “clientGenerator”. The “wsdlurl2path” method returns the path of the generated client stub file.

→ Gets the list of operations using the method “path2ops” in module creatorEngineComplex.py in package clientGenerator package.

→ Print all the operations and web service info to the Galaxy output dataset in a tab-delimited format. This data is used as input to the next step.

Galaxy output after Step a
Column 1 – list of operations
Column 2 – web srvice path
Column 3 – WSDL url

getWADLMethods() :
→ Start JVM using jpype
→ With the help of JVM invoke the WADL parser coded in java to parse the WADL file.
→ The parser parses the WADL document and gives the list of operations (i.e. resource urls)
→ Write all the operations and web service info to the Galaxy output dataset. This data is just displayed as in case of getWSDLMethods().

getSAWADLMethods()
Works just like getWADLMethods()

Step b :
WebServiceTool_input_method_REST_SOAP.xml->WebServiceTool_input_method_m.py

The XMLToolfile, WebServiceTool_input_method_REST_SOAP.xml specifies three input parameters.

- The previous step: A drop-down that holds the history of all the steps performed in Galaxy, with the step immediately performed before as pre-selected .
- The web service chosen: Displays the Web service tool chosen in Step 1.
- Operation of the web service to add as tool

WebServiceTool_input_method_REST_SOAP.xml invokes python module WebService_input_method_m.py

This module checks if the WebService is REST or SOAP
If web service is SOAP:
For every operation selected it will call wsdlClient() method in generateClient.py for workflow clients and generateClient1.py in case of stand-alone clients, if they are not already added to Galaxy with the help of the method isToolPresent().

If web service is REST:
For every operation selected it will call wadlClient() method in generateClient.py for workflow clients and generateClient1.py in case of stand-alone clients, if they are not already added to Galaxy with the help of the methodd isToolPresent().

Module : generateClient.py

This module is used to create the tool xml files for the webservice operation. The xml file is created in the program line by line. The tool cretaed is stored in $GALAXY_HOME/tools/WebService_REST_SOAP/workflowclients. The python file corresponding to the xml file generated here is client_1.py stored in $GALAXY_HOME/tools/WebServiceToolEWorkflow_REST_SOAP/workflowclients.

Methods
isToolRequired()

This method return true if the tool(web service operation) is already present in Galaxy else it returns false. It opens the tool_conf.xml file and loops through the "Web Service Workflow Tools" sections. Then opens each and every tool (xml file) and looks for the description tag for the "Web Service" and "Client for Method" values and then checks if it same as the opearation and web service. As the code is dependent on description tag value of the toool xml file, any change in the description tag value of the tool xml file can impact this function.

wsdlClient()

This method creates the tool xml file for the web service operation to be used in workflow. The client created only displays the required parametrs. The client also has an option to show the optional parameters which the user can enter if he wants.
As galaxy required that every tool shoul have an unique identifier we make the id for the client as client_x where x is a number that is incremented each time when a tool is created. This number is stored in clientcount.xml under the clients folder. The tools is created according to the input paramters of the webservcie operation.

wadlClient()

This method creates the tool xml file for the web service operation to be used in workflow. The client created only displays the required parametrs. The client also has an option to show the optional paramters which the user can enter if he wants.
As galaxy required that every tool shoul have an unique identifier we make the id for the client as client_x where x is a number that is incremented each time when a tool is created. This number is stored in clientcount.xml under the clients folder. The tools is created according to the input paramters of the webservcie operation.

Module : generateClient1.py

This has the same functionality compared to that of generateClient.py except that it tool that this module cretaes is used for invoking web services as stand alone. The client files are stores in $GALAXY_HOME/tools/WebServiceToolEWorkflow_REST_SOAP/clients.

The python file corresponding to the xml file generated here is client_1.py stored in $GALAXY_HOME/tools/WebServiceToolEWorkflow_REST_SOAP/clients.

Module : client_1.py

This module is used for invoking the web service. It can invoke either a SOAP or REST Web service. In case of SOAP web service it makes use of ZSI. And for REST web services it just make use of the http request.