ENVI Py for ArcGIS Library Documentation

Release 1.0

Exelis Visual Information Solutions, Inc.

Aug 16, 2017

Contents

1 Usage **3**
1.1 Create GPToolbox 3
1.2 Test GPToolbox Data Types 4

2 API Documentation **5**
2.1 GPToolbox 5
2.2 GPTool Parameter Builder 6
2.3 GPTool Parameter Template 7

Python Module Index **9**

ENVI Py for ArcGIS Library provides tools for generating ArcGIS python toolboxes.

See http://www.harrisgeospatial.com/ for more details on product offerings.

CHAPTER 1

Usage

Create GPToolbox

To generate an empty toolbox from python, use the GPToolbox class and initialize with an empty list of task objects:

```
from envipyarclib import GPToolbox
toolbox = GPToolbox([], 'DEMO')
toolbox.create_toolbox('C:\\TEMP\\demo.pyt')
```

To create a toolbox with empty GPTools you must first create a Task class defining the properties the GPToolbox supports:

```
class Task(object):

    def __init__(self, name, display_name, description):
        self.name = name
        self.display_name = display_name
        self.description = description
        self.uri = name

    @property
    def parameters(self):
        return dict()
```

Once the Task class is defined you can pass in task objects to the GPToolbox and generate empty GPtools:

```
toolbox = GPToolbox([Task('firstTask', 'First Task', 'Does nothing'),
                     Task('secondTask', 'Second Task', 'Still Does nothing')],
                     'DEMO')
toolbox.create_toolbox('C:\\TEMP\\demo.pyt')
```

To import python modules or any global functionality to the toolbox, you can use the imports_template keyword when initializing the GPToolbox. The template string must be formatted with zero indentation. For example to inlcude arcpy and os:

```
from string import Template
from envipyarclib import GPToolbox

imports_template = Template('''
import os
import arcpy
''')
```

```
toolbox = GPToolbox([Task('firstTask', 'First Task', 'Does nothing')], 'DEMO',
                    imports_template=imports_template)
toolbox.create_toolbox('C:\\TEMP\\demo.pyt')
```

To add code to the execute method of all GPTools, you can use the execute_template keyword when initializing the GPToolbox. The template string must be formatted with 2 indentations. From our previous exmample, to add the system path to the messages when running the GPTool:

```
from string import Template
from envipyarclib import GPToolbox

imports_template = Template('''
import os
import arcpy
''')

execute_template = Template('''
        messages.AddMessage('System Path: ' + str(os.sys.path))
''')

toolbox = GPToolbox([Task('firstTask', 'First Task', 'Does nothing')], 'DEMO',
                    imports_template=imports_template,
                    execute_template=execute_template)
toolbox.create_toolbox('C:\\TEMP\\demo.pyt')
```

Test GPToolbox Data Types

This library provides test cases and test tasks for testing ENVI and IDL data types available in the envipyarclib.test package. To test a data type, you must implement the test config abstract base class for generating a toolbox and importing it into arcpy before running a test:

```
import arcpy
from envipyarclib.test.config import Config

class MyConfig(Config):

    def __init__
    def setup_toolbox(self, engine_name, task_name, toolbox_name):
        # Create python toolbox here
        arcpy.ImportToolbox(toolbox_file)
```

Once the Config class is created you can import a data type test in your module for testing. Once imported, the unittest module will be able to find and run the test cases:

```
from envipyarclib.test.datatype.bool import TestDataTypeBool

config = MyConfig()

# Attach the config to the test case
TestDataTypeBool.config = config
```

CHAPTER 2

API Documentation

GPToolbox

GPToolbox is used as a base class to create GPTool wrappers for ENVI, IDL, and GSF Analytics.

class envipyarclib.gptoolbox.**GPToolbox**(*tasks=None*, *alias=None*, *imports_template=<string.Template object>*, *execute_template=<string.Template object>*, *parameter_templates=None*)

GPToolbox is used as a base class to create GPTool wrappers for ENVI, IDL, and GSF Analytics.

Parameters

- **tasks** – a list of tasks to map to GPTools where each task name is a GPTool in the toolbox.
- **alias** – The alias of the generated toolbox
- **imports_template** – The template string code for defining imports
- **execute_template** – The template string code for GPTool execution
- **parameter_templates** – The python package containing parameter templates. Templates must implement the envipyarclib.gptool.parameter.Template class.

create_tool(*task*)

Creates a new GPTool for the toolbox.

create_toolbox(*filename*)

Creates a new Python toolbox where each task name is a GPTool in the toolbox.

Parameters

- **filename** – the filename of the generated toolbox
- **service_name** – The name of the ESE service containing the tasks. Only tasks from one service may be used.
- **tasks** – The list of tasks from the service to build as GPTools.

import_script(*script_name*)

Finds the script file and copies it into the toolbox

GPTool Parameter Builder

The GPTool Parameter Builder is responsible for creating code blocks to be placed into the main GPTool template. Each create method corresponds to a GPTool method that must be defined in order for it to be a valid tool. These methods implement the GetParameterInfo, UpdateParameter, and Execute methods of the GPTool Python class.

For example, to define a GPTool in a Python toolbox, start out with a class definition and implement the methods arcpy expects:

```
class myTool(Object):
    def __init__(self):
        self.label = "My Tool"
        self.description = "Tool description"
        self.canRunInBackground = True

    def getParameterInfo(self):
        # builder.create_param_info() goes here

    def is Licensed(self):
        return True

    def updateParameters(self, parameters):
        # builder.create_update_parameter goes here

    def updateMessages(self, parameters):
        return

    def execute(self, parameters, messages):
        # builder.create_pre_execute goes here
        # submit job
        # builder.create_post_execute goes here
```

class envipyarclib.gptool.parameter.builder.**ParameterMap**
Convenience class for holding a template map

load_default_templates()
Load the default templates

register_template(*module*)
Register a non-default template

Parameters **module** – The full package path including the module name of the template to load.

exception envipyarclib.gptool.parameter.builder.**UnknownDataTypeError**
Error class for raising unknown datatypes

envipyarclib.gptool.parameter.builder.**convert_list**(*in_list*)
Converts a list of strings to a printable list of object names

envipyarclib.gptool.parameter.builder.**create_param_info**(*task_params*, *parameter_map*)
Builds the code block for the GPTool GetParameterInfo method based on the input task_params.

Parameters **task_params** – A list of task parameters to map to GPTool parameters.

Returns A string representing the code block to the GPTool GetParameterInfo method.

envipyarclib.gptool.parameter.builder.**create_post_execute**(*task_params*, *parameter_map*)
Builds the code block for the GPTool Execute method after the job is submitted based on the input task_params.

Parameters **task_params** – A list of task parameters from the task info structure.

Returns A string representing the code block to the GPTool Execute method.

envipyarclib.gptool.parameter.builder.**create_pre_execute**(*task_params*, *parameter_map*)

Builds the code block for the GPTool Execute method before the job is submitted based on the input task_params.

Parameters **task_params** – A list of task parameters from the task info structure.

Returns A string representing the code block to the GPTool Execute method.

envipyarclib.gptool.parameter.builder.**create_update_parameter**(*task_params*, *parameter_map*)

Builds the code block for the GPTool UpdateParameter method based on the input task_params.

Parameters **task_params** – A list of task parameters from the task info structure.

Returns A string representing the code block to the GPTool UpdateParameter method.

GPTool Parameter Template

class envipyarclib.gptool.parameter.template.**Template**(*data_type*)

Interface class for mapping Task parameters to ArcGIS GPTool parameters.

default_value()

Defines the code block for this parameter data type in the GPTool GetParameterInfo if a default value exists.

Returns Returns the string.Template object.

get_parameter(*task_param*)

Defines the code block for this parameter data type in the GPTool GetParameterInfo method. All code returned must begin with 2 indents. The template is substituted against the GP parameter dictionary.

Parameters **task_param** – The task parameter information.

Returns Returns the string.Template object.

parameter_names(*task_param*)

Defines the code block for the parameter variable names in the GPTool GetParameterInfo method.

Parameters **task_param** – The task parameter

Returns A list of string.Template objects representing the parameter variable names defined in get_parameter.

post_execute()

Defines the code block for this parameter data type in the GPTool Execute method after the task is executed.

Returns Returns the the string.Template object

pre_execute()

Defines the code block for this parameter data type in the GPTool Execute method before the task is executed.

Returns Returns the string.Template object

update_parameter()

Defines the code block for this parameter data type in the GPTool UpdateParameter method.

Returns Returns the string.Template object.

e

envipyarclib.gptool.parameter.builder, 6
envipyarclib.gptool.parameter.template, 7
envipyarclib.gptoolbox, 5

C

convert_list() (in module envipyarclib.gptool.parameter.builder), 6
create_param_info() (in module envipyarclib.gptool.parameter.builder), 6
create_post_execute() (in module envipyarclib.gptool.parameter.builder), 6
create_pre_execute() (in module envipyarclib.gptool.parameter.builder), 7
create_tool() (envipyarclib.gptoolbox.GPToolbox method), 5
create_toolbox() (envipyarclib.gptoolbox.GPToolbox method), 5
create_update_parameter() (in module envipyarclib.gptool.parameter.builder), 7

D

default_value() (envipyarclib.gptool.parameter.template.Template method), 7

E

envipyarclib.gptool.parameter.builder (module), 6
envipyarclib.gptool.parameter.template (module), 7
envipyarclib.gptoolbox (module), 5

G

get_parameter() (envipyarclib.gptool.parameter.template.Template method), 7
GPToolbox (class in envipyarclib.gptoolbox), 5

I

import_script() (envipyarclib.gptoolbox.GPToolbox method), 5

L

load_default_templates() (envipyarclib.gptool.parameter.builder.ParameterMap method), 6

P

parameter_names() (envipyarclib.gptool.parameter.template.Template method), 7
ParameterMap (class in envipyarclib.gptool.parameter.builder), 6
post_execute() (envipyarclib.gptool.parameter.template.Template method), 7
pre_execute() (envipyarclib.gptool.parameter.template.Template method), 7

R

register_template() (envipyarclib.gptool.parameter.builder.ParameterMap method), 6

T

Template (class in envipyarclib.gptool.parameter.template), 7

U

UnknownDataTypeError, 6
update_parameter() (envipyarclib.gptool.parameter.template.Template method), 7