Setting up a WPS process using PyWPS

Using PyWPS as a WPS server instance allows you to easily transform about any Python into a WPS process. In this example we show you which code has to be added to your script so that it is recognized by PyWPS.

The Process

To demonstrate the set up of a PyWPS process we show you how to convert a (very) simple piece of Python code into a PyWPS process. The process in the code box below uses two input values which are added together.

```
# Input
val_1 = 2
val_2 = 6


# Process
result = val_1 + val_2


# Output
print result
```

Add PyWPS Metadata

PyWPS requires some metadata about different components of the process. In the script a process identifier, title and abstract can be added as well as a definition of the different in- and output parameters.

```
class Process(WPSProcess):
    def __init__(self):

        WPSProcess.__init__(self,
             identifier       = "inputs_added",
             title            = "This process adds the inputvalues",
             version          = "1",
             storeSupported   = "false",
             statusSupported  = "false",
             abstract         = "test process")
```

Tag	Explanation
identifier	Short name for the process, this is also used in the request URL
title	(Long) Title of the process
version	Version of the process
storeSupported	Indicate whether storage of results is supported (true or false)
statusSupported	Indicate whether status updates are supported (true or false)
abstract	Abstract in which for example an explanation on how to use the process is given

Defining PyWPS in- and outputs

In our example script we have two input variables and one output variable. We have to define these variables so PyWPS knows what to expect in the request. This in- and output information, together with the meta information from the previous section is published at the XML document returned fromt he DescribeProcess request.

```
        self.val_1  = self.addLiteralInput(identifier  = "val_1",
                                           title       = "Input Value 1",
                                           type        = FloatType)

        self.val_2  = self.addLiteralInput(identifier  = "val_2",
                                           title       = "Input Value 2",
                                           type        = FloatType)

        self.result = self.addLiteralOutput(identifier = "result",
                                           title       = "Value of the process result",
                                           type        = FloatType)
```

In this case the input parameters are 'Literal' in- and output, this means that it is a value (float or integer) or a string given directly in the request URL. In case of a file such as a NetCDF-, TXT- or Shapefile the term 'Complex' in- and output is used.

Tag	Explanation
identifier	Short name for the variable, this is also used in the request URL
title	(Long) Title for the variable.
type	Type of the variable (float, integer, string, etc.), don't forget to 'import' the type in your script

The complete script

To complete the total script the met the process part has to be adjusted to fit the in- and outputs and has to be stated under the 'def exectute(self):' code. See the code box below for the complete script.

```
from pywps.Process import WPSProcess
from types import FloatType

class Process(WPSProcess):
     def __init__(self):

         WPSProcess.__init__(self,
              identifier       = "inputs_added",
              title            = "This process adds the inputvalues",
              version          = "1",
              storeSupported   = "false",
              statusSupported  = "false",
              abstract         = "test process")

         self.val_1  = self.addLiteralInput(identifier  = "val_1",
                                            title       = "Input Value 1",
                                            type        = FloatType)

         self.val_2  = self.addLiteralInput(identifier  = "val_2",
                                            title       = "Input Value 2",
                                            type        = FloatType)

         self.result = self.addLiteralOutput(identifier = "result",
                                            title       = "Value of the process result",
                                            type        = FloatType)

     def execute(self):
         # Input
         val_1 = self.val_1.getValue()
         val_2 = self.val_2.getValue()

         # Process
         result = val_1 + val_2

         # Output
         self.result.setValue(result)
         return
```

Add the process to your WPS server

To get the process actually working on the WPS server some final steps have to be made. First is to upload the script to the process folder of your PyWPS server, make sure that the filename is equal to the process identifier you defined in the initialization part. After uploading the configuration file (*__init__*.py) has to be updated, simply add the new process to the list.

```
__all__ = ["simple_buffer","constituents","IDT_simple","inputs_added"]
```

Afterwards the server instance has to be restarted to get the new process available from the web.