1 Contents: 3
1.1 Flask .................................................. 3
1.2 Configuration ........................................... 5
1.3 Demo processes ......................................... 6
1.4 File structure of PyWPS-Demo .......................... 6
1.5 Indices and tables ....................................... 6
PyWPS-Demo is little project, which shall be distributed along with PyWPS project. This is just demo server instance of PyWPS, with several example processes. We are adding also sample demo.py which can be used with the Flask microframework.

For more comprehensive documentation visit http://pywps.readthedocs.io/en/latest/
1.1 Flask

Flask is a microframework for web applications in Python. Some characteristics of Flask:

• built-in development server and debugger
• RESTful request dispatching
• 100% WSGI 1.0 compliant

You can develop your PyWPS application and modules using a local Flask server and then move it to a production environment (e.g. with Apache2 HTTP server).

1.1.1 Start PyWPS server

Start the PyWPS demo-server using Flask’s built-in server:

```bash
$ python3 demo.py
```

You should see some output from the WPS-server that is now running at http://localhost:5000/wps. Alternatively you may use Python2 and issue `python demo.py`.

1.1.2 Testing the server

Basics

You should be able to interact with the WPS-server like any other HTTP-server, i.e. either requesting URLs using your web browser or using commandline tools like `wget` or `curl`. For example using `wget` to fetch the Capabilities of the WPS Server:

```bash
$ wget --content-on-error -O - "http://localhost:5000/wps?service=wps&request=getcapabilities"
```

Or visit the URL directly in the browser:

```plaintext
http://localhost:5000/wps?service=wps&request=getcapabilities
```

In both cases you should see the response:
If anything goes wrong, you should see the result in Flask terminal, for example:

http://localhost:5000/wps

With response:

<?xml version="1.0" encoding="UTF-8"?>
<!-- PyWPS 4.0.0-beta1 -->
<ows:Exception exceptionCode="MissingParameterValue" locator="service"/>

And output from Flask in the terminal:

ERROR:PYWPS:Exception: code: 400, locator: service, description: service
NoneType

Processes

The GetCapabilities response in the previous section lists the WPS Processes available on the WPS demo-server.

Issue a DescribeProcess WPS request for the say_hello WPS Process using the URL:

http://127.0.0.1:5000/wps?service=WPS&request=DescribeProcess&version=1.0.0&identifier=say_hello

Note that the version parameter is required with most WPS-requests. The output includes the Inputs for this WPS Process:

<wps:ProcessDescriptions xmlns:wps="http://www.opengis.net/wps/1.0.0">
  <ProcessDescription wps:processVersion="1.3.3.7" storeSupported="true" statusSupported="true">
    <ows:Identifier>say_hello</ows:Identifier>
    <ows:Title>Process Say Hello</ows:Title>
    <DataInputs>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>name</ows:Identifier>
        <ows:Title>Input name</ows:Title>
        <LiteralData>
          <ows:DataType ows:reference="urn:ogc:def:dataType:OGC:1.1:string">string</ows:DataType>
          <ows:AnyValue/>
        </LiteralData>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>response</ows:Identifier>
        <ows:Title>Output response</ows:Title>
        <LiteralOutput>
          <ows:DataType ows:reference="urn:ogc:def:dataType:OGC:1.1:string">string</ows:DataType>
        </LiteralOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</wps:ProcessDescriptions>
This response indicates that the say_hello WPS Process requires one parameter name. Execute the say_hello WPS Process with the URL:

```
http://127.0.0.1:5000/wps?service=WPS&request=Execute&version=1.0.0&identifier=say_hello&datainputs=name=Luis
```

You should see a response like:

```
<wps:ExecuteResponse xmlns:wps="http://www.opengis.net/wps/1.0.0"
    service="WPS" version="1.0.0" xml:lang="en-US"
    serviceInstance="http://localhost:5000/wps?service=WPS&amp;request=GetCapabilities"
    statusLocation="http://localhost:5000/outputs/50a071eb-6d21-11e6-9dd5-9801a7996b55.xml">
    <wps:Process wps:processVersion="1.3.3.7">
        <ows:Identifier>say_hello</ows:Identifier>
        <ows:Title>Process Say Hello</ows:Title>
    </wps:Process>
    <wps:Status creationTime="2016-08-28T15:14:13Z">
        <wps:ProcessSucceeded>PyWPS Process finished</wps:ProcessSucceeded>
    </wps:Status>
    <wps:ProcessOutputs>
        <wps:Output>
            <ows:Identifier>response</ows:Identifier>
            <ows:Title>Output response</ows:Title>
            <wps:Data>
                <wps:LiteralData dataType="urn:ogc:def:dataType:OGC:1.1:string"
                    uom="urn:ogc:def:uom:OGC:1.0:unity">Hello Luis</wps:LiteralData>
            </wps:Data>
        </wps:Output>
    </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

NB it is recommended to use HTTP POST requests for invoking WPS Execute operations as normally DataInputs will be more complex.

### 1.2 Configuration

PyWPS-Demo comes with configuration file, which shall work for both - Flask and Apache2 deployment. It's stored in pywps.cfg some default values. You are advised to play with the configuration values and see what they do. More detailed documentation about PyWPS configuration can be found at http://pywps.readthedocs.io/en/latest/configuration.html

Also have a look at File structure of PyWPS-Demo chapter, which describes file and directory structure of PyWPS-Demo.
1.3 Demo processes

PyWPS-Demo comes along with sample processes, so you could get inspired how to write the process:

```python
class processes.area.Area
class processes.bboxinout.Box
class processes.buffer.Buffer
class processes.centroids.Centroids
class processes.feature_count.FeatureCount
class processes.grassbuffer.GrassBuffer
class processes.sayhello.SayHello
class processes.sleep.Sleep
class processes.ultimate_question.UltimateQuestion
```

1.4 File structure of PyWPS-Demo

This chapter describes files and directories structure of PyWPS-Demo and their relationship to `Configuration`

demo.py
pywps.cfg
processes
logs
outputs
workdir
static

Todo
Add some text :-)
Index

A
Area (class in processes.area), 6

B
Box (class in processes.bboxinout), 6
Buffer (class in processes.buffer), 6

C
Centroids (class in processes.centroids), 6

F
FeatureCount (class in processes.feature_count), 6

G
GrassBuffer (class in processes.grassbuffer), 6

S
SayHello (class in processes.sayhello), 6
Sleep (class in processes.sleep), 6

U
UltimateQuestion (class in processes.ultimate_question), 6