Matlab WPS convention

This page describes the convention for the PyWPS plugin to find and expose matlab processes.

Example

oil_spill.m

```
function [data] = oil_spill(track, volume)
% OIL_SPILL generate a dataset of an oil spill based on a track and total volume
%
% This is also part of the abstract
%
% input:
% track = linestring
% volume = double
%
% output:
% data = file.nc
%
```

Finding processes

Matlab functions can be exposed as WPS processes. The processes are found by putting matlab scripts in the directory that is exposed by the environment variable **MATLAB_PROCESSES** (analogue to **PYWPS_PROCESSES**). Matlab processes have the form of a matlab function, stored within a file with the name **function.m**.

WPS Process properties

The WPS process properties can be specified according to the table below. A process is a function and the properties are introspected from the comments by the following convention.

WPS property	Matlab implementation	example
identifier	function name, lower case	run_model -> run_model
title	function name, Title cased, _ replaced by spaces	run_model -> Run model
abstract	first comment line, after CAPITALIZED function name and following comment block	% FUNCTION some text -> some text
metadata	not supported	
profile	not supported	
wsdl	not supported	
version	not supported	
service	always WPS	
language	always en-GB	
DataInputs	list below % input:	see example and details below
ProcessOutputs	list below % output:	see example and details below
storeSupported	always false	
statusSupported	always false	

WPS Input properties

The input properties are the function arguments, which properties are introspected from the comments by the following convention.

WPS property	Matlab implementation	example

identifier	function name, lower case	run_model -> run_model
title	function name, Title cased, _ replaced by spaces	run_model -> Run model
abstract	first comment line, after CAPITALIZED function name and following comment block	% FUNCTION some text -> some text
minOccurs	always 1	
maxOccurs	always 1	
Metadata	not supported	
InputForm Choice	not supported	
ComplexData	SFS types supported, by WKT type: linestring,geometry,point,etc...	
LiteralData	basic types supported: float,int,string	
BoundingBox Data	specified by the word bbox	

WPS Matlab JSON input.

Because you can't call matlab functions dynamically we prepare input for the matlab processes in JSON messages.
These JSON messages have the following form.

```
// The following is an example of the CouchDB job queue
// The _id, _rev and _attachments are couchdb specific
{
   "_id": "723cd54245f4c578550768758b001564",
   "_rev": "6-e16ee53ee4642e2a3a080306f727a43f",
   "identifier": "tide_analysis",
   "dataInputs": {
       "timeseries": "data.xls",
       "location": "POINT(52 3)"
   },
   "_attachments": {
       "data.xls": {
           "content_type": "application/vnd.ms-excel",
           "revpos": 5,
           "length": 50688,
           "stub": true
       }
   }
```

WPS Matlab JSON input.

- install CouchDB from http://couchdb.apache.org/
- Via the `Create Database` link create a database called 'wps' (small case)
- Click on wps database,
- Via the `New Document` link create a new view.
- Manually update the source block (double click in the source block) by adding the following code:

```
{
   "_id": "_design/views",
   "language": "javascript",
   "views": {
       "processes": {
           "map": "function(doc) {\n\n  if (doc.type == \"processes\") {\n  \n  doc.processes.forEach
(function(element){\n  emit(element.identifier, element);\n  });\n}\n}"
       },
       "input": {
           "map": "function(doc) {\n  if (doc.type == \"input\") {\n     emit(null, doc);\n}\n}"
       },
       "output": {
           "map": "function(doc) {\n  if (doc.type == \"output\") {\n     emit(doc._id, doc);\n}\n}"
       },
       "matlab": {
           "map": "function(doc) {\n\n  if (doc.type == \"processes\" && doc.language == \"matlab\")
{\n  \n  \n   emit(doc._id, doc);\n  }\n}"
       }
   }
}
```

- If you modify an existing document, keep the original `"_rev": "BUT KEEP THE AUTOMATIC REV CODE GENERATED BY YOUR COUCHDB"`
- Test `http://localhost:5984/wps/_design/views/_view/matlab` to look like:

```
{"total_rows":0,"offset":0,"rows":[

]}
```

- Switch to `queue_url = 'http://localhost:5984'` in the matlab wps runner `wps.runner.run`
- Check out this fork of pywps https://github.com/openearth/PyWPS and use branch couchprocesses
- configure your couchdb server in https://github.com/openearth/PyWPS/blob/couchprocesses/pywps/default.cfg