
Emu Documentation

Release 0.5

Birdhouse

May 08, 2017

1	Installation	3
1.1	Non-default installation	4
1.2	Run Emu as Docker container	4
2	Configuration	5
3	Processes	7
4	Tutorial: using Docker	9
4.1	Using docker-compose	10
5	Sphinx AutoAPI Index	11
5.1	emu	11
5.2	emu.processes	11
5.3	emu.processes.wps_bbox	11
5.4	emu.processes.wps_binaryoperator	11
5.5	emu.processes.wps_chomsky	12
5.6	emu.processes.wps_dummy	12
5.7	emu.processes.wps_error	13
5.8	emu.processes.wps_inout	13
5.9	emu.processes.wps_multiple_outputs	13
5.10	emu.processes.wps_nap	14
5.11	emu.processes.wps_say_hello	14
5.12	emu.processes.wps_sleep	14
5.13	emu.processes.wps_ultimate_question	14
5.14	emu.processes.wps_wordcounter	15
5.15	emu.tests	15
5.16	emu.tests.common	15
5.17	emu.tests.test_wps_bbox	15
5.18	emu.tests.test_wps_caps	15
5.19	emu.tests.test_wps_chomsky	16
5.20	emu.tests.test_wps_dummy	16
5.21	emu.tests.test_wps_hello	16
5.22	emu.tests.test_wps_inout	16
5.23	emu.tests.test_wps_ultimate_question	16
5.24	emu.tests.test_wps_wordcounter	16

Emu (the bird) *Emus are curious birds who are known to follow and watch other animals and humans. Emus do not sleep continuously at night but in several short stints sitting down. [..].* ([Wikipedia](#)).

Emu is a Python package with some test process for Web Processing Services (WPS). Currently it is using the [PyWPS-4](#) server.

Emu is part of the [Birdhouse](#) project.

CHAPTER 1

Installation

The installation is using the Python distribution system [Anaconda](#) to maintain software dependencies. Anaconda will be installed during the installation process in your home directory `~/anaconda`.

The installation process setups a conda environment named `emu` with all dependent conda (and pip) packages. The installation folder (for configuration files etc) is by default `~/birdhouse`. Configuration options can be overridden in the buildout `custom.cfg` file. The `ANACONDA_HOME` and `CONDA_ENVS_DIR` locations can be changed in the `Makefile.config` file.

The default installation *does not need admin rights* and files will only be written into the `$HOME` folder of the installation user. The services are started using [supervisor](#) and run as the installation user.

Now, check out the `emu` code from GitHub and start the installation:

```
$ git clone https://github.com/bird-house/emu.git
$ cd emu
$ make clean install
```

After successful installation you need to start the services:

```
$ make start # starts supervisor services
$ make status # shows supervisor status
```

The depolyed WPS service is by default available on <http://localhost:8094/wps?service=WPS&version=1.0.0&request=GetCapabilities>.

Check the log files for errors:

```
$ tail -f ~/birdhouse/var/log/pywps/emu.log
$ tail -f ~/birdhouse/var/log/supervisor/emu.log
```

You will find more information about the installation in the [Makefile documentation](#).

Non-default installation

You can customize the installation to use different ports, locations and run user.

To change the anaconda location edit the `Makefile.config`, for example:

```
ANACONDA_HOME ?= /opt/anaconda
CONDA_ENVS_DIR ?= /opt/anaconda/envs
```

You can install emu as root and run it as unprivileged user like `www-data`:

```
root$ mkdir -p /opt/birdhouse/src
root$ cd /opt/birdhouse/src
root$ git clone https://github.com/bird-house/emu.git
root$ cd emu
```

Edit `custom.cfg`:

```
[buildout]
extends = buildout.cfg

[settings]
hostname = emu
http-port = 80
output-port = 500
log-level = WARN

# deployment options
prefix = /opt/birdhouse
user = www-data
etc-user = root
```

Run the installation and start the services:

```
root$ make clean install
root$ make start          # stop or restart
root$ make status
```

Run Emu as Docker container

You can also run Emu as a Docker container, see the [Tutorial](#).

CHAPTER 2

Configuration

If you want to run on a different hostname or port then change the default values in `custom.cfg`:

```
$ cd emu
$ vim custom.cfg
$ cat custom.cfg
[settings]
hostname = localhost
http-port = 8094
```

After any change to your `custom.cfg` you **need** to run `make update` again and restart the supervisor service:

```
$ make update    # or install
$ make restart
```


CHAPTER 3

Processes

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

- `emu.processes.wps_say_hello.SayHello`
- `emu.processes.wps_sleep.Sleep`

CHAPTER 4

Tutorial: using Docker

Emu WPS is available as docker image. You can download the docker image from [DockerHub](#) or build it from the provided Dockerfile.

Start the container with the following command:

```
$ docker run -i -d -p 8080:8080 -p 8000:8000 -p 9001:9001 --name=emu birdhouse/emu
```

The ports are:

- PyWPS port: 8080
- NGINX file service port for the outputs: 8000
- Supervisor port: 9001 (optional)

You can map the container port also to another port on your machine, for example: `-p 8094:8080` (your machine port=8094, container port=8080).

Check the docker logs:

```
$ docker logs emu
```

Show running docker containers:

```
$ docker ps
```

Run a GetCapabilities WPS request:

```
http://localhost:8080/wps?service=WPS&version=1.0.0&request=getcapabilities
```

Run DescribeProcess WPS request for *Hello*:

```
http://localhost:8080/wps?service=WPS&version=1.0.0&request=describeprocess&identifier=hello
```

Execute *Hello* process with you user name:

```
http://localhost:8080/wps?service=WPS&version=1.0.0&request=execute&identifier=hello&DataInputs=name=Pingu
```

Install *Birdy* WPS command line tool from Anaconda (Anaconda needs to be installed and in your PATH):

```
$ conda install -c birdhouse birdhouse-birdy
```

Use Birdy to access Emu WPS service:

```
$ export WPS_SERVICE=http://localhost:8080/wps
$ birdy -h
$ birdy hello -h
$ birdy hello --name Pingu
```

Stop and remove docker container:

```
$ docker stop emu_wps
$ docker rm emu_wps
```

Using docker-compose

Use `docker-compose` (you need a recent version > 1.7) to start the container:

```
$ git clone https://github.com/bird-house/emu.git
$ cd emu
$ docker-compose up -d
$ docker-compose logs emu
```

You can change the ports and hostname with environment variables:

```
$ HOSTNAME=emu HTTP_PORT=8094 SUPERVISOR_PORT=48094 docker-compose up
```

Now the WPS is available on port 8094: <http://emu:8094/wps?service=WPS&version=1.0.0&request=GetCapabilities>.

You can also customize the `docker-compose.yml` file. See the [docker-compose documentation](#).

This page is the top-level of your generated API documentation. Below is a list of all items that are documented here.

emu

emu.processes

emu.processes.wps_bbox

Box

`class emu.processes.wps_bbox.Box`

Imports

•<UNKNOWN>

`_handler` (*self, request, response*)

emu.processes.wps_binaryoperator

BinaryOperator

`class emu.processes.wps_binaryoperator.BinaryOperator`

Imports

•<UNKNOWN>

`_handler` (*self, request, response*)

emu.processes.wps_chomsky

Chomsky

`class emu.processes.wps_chomsky.Chomsky`

Imports

•<UNKNOWN>

Summary

Generates a random chomsky text: <http://code.activestate.com/recipes/440546-chomsky-random-text-generator/>

CHOMSKY is an aid to writing linguistic papers in the style of the great master. It is based on selected phrases taken from actual books and articles written by Noam Chomsky. Upon request, it assembles the phrases in the elegant stylistic patterns that Chomsky is noted for. To generate n sentences of linguistic wisdom, type

(CHOMSKY n) – for example (CHOMSKY 5) generates half a screen of linguistic truth.

`_handler` (*self, request, response*)

emu.processes.wps_dummy

Summary

DummyProcess to check the WPS structure

Author: Jorge de Jesus (jorge.jesus@gmail.com) as suggested by Kor de Jong

Dummy

`class emu.processes.wps_dummy.Dummy`

Imports

•<UNKNOWN>

`_handler` (*self, request, response*)

emu.processes.wps_error

ShowError

```
class emu.processes.wps_error.ShowError
```

Imports

- <UNKNOWN>

```
_handler (self, request, response)
```

emu.processes.wps_inout

InOut

```
class emu.processes.wps_inout.InOut
```

Imports

- <UNKNOWN>

Summary

This process defines several types of literal type of in- and outputs.

TODO: add literal input with value range[(0,100)] ... see pywps doc

```
_handler (self, request, response)
```

emu.processes.wps_multiple_outputs

MultipleOutputs

```
class emu.processes.wps_multiple_outputs.MultipleOutputs
```

Imports

- <UNKNOWN>

```
_handler (self, request, response)
```

emu.processes.wps_nap

Nap

```
class emu.processes.wps_nap.Nap
```

Imports

- <UNKNOWN>

```
__handler (self, request, response)
```

emu.processes.wps_say_hello

SayHello

```
class emu.processes.wps_say_hello.SayHello
```

Imports

- <UNKNOWN>

```
__handler (self, request, response)
```

emu.processes.wps_sleep

Sleep

```
class emu.processes.wps_sleep.Sleep
```

Imports

- <UNKNOWN>

```
__handler (self, request, response)
```

emu.processes.wps_ultimate_question

UltimateQuestion

```
class emu.processes.wps_ultimate_question.UltimateQuestion
```

Imports

•<UNKNOWN>

`_handler` (*self*, *request*, *response*)

emu.processes.wps_wordcounter

WordCounter

`class emu.processes.wps_wordcounter.WordCounter`

Imports

•<UNKNOWN>

`_handler` (*self*, *request*, *response*)

emu.tests

emu.tests.common

`emu.tests.common.client_for` (*service*)

WpsTestClient

`class emu.tests.common.WpsTestClient`

Imports

•<UNKNOWN>

`get` (*self*)

emu.tests.test_wps_bbox

`emu.tests.test_wps_bbox.test_wps_bbox` ()

emu.tests.test_wps_caps

`emu.tests.test_wps_caps.test_wps_caps` ()

emu.tests.test_wps_chomsky

```
emu.tests.test_wps_chomsky.test_wps_chomsky()
```

emu.tests.test_wps_dummy

```
emu.tests.test_wps_dummy.test_wps_dummy()
```

emu.tests.test_wps_hello

```
emu.tests.test_wps_hello.test_wps_hello()
```

emu.tests.test_wps_inout

```
emu.tests.test_wps_inout.test_wps_inout()
```

emu.tests.test_wps_ultimate_question

```
emu.tests.test_wps_ultimate_question.test_wps_ultimate_question()
```

emu.tests.test_wps_wordcounter

```
emu.tests.test_wps_wordcounter.test_wps_wordcount()
```

e

- [emu](#), 11
- [emu.processes](#), 11
 - [emu.processes.wps_bbox](#), 11
 - [emu.processes.wps_binaryoperator](#), 11
 - [emu.processes.wps_chomsky](#), 12
 - [emu.processes.wps_dummy](#), 12
 - [emu.processes.wps_error](#), 13
 - [emu.processes.wps_inout](#), 13
 - [emu.processes.wps_multiple_outputs](#), 13
 - [emu.processes.wps_nap](#), 14
 - [emu.processes.wps_say_hello](#), 14
 - [emu.processes.wps_sleep](#), 14
 - [emu.processes.wps_ultimate_question](#), 14
 - [emu.processes.wps_wordcounter](#), 15
- [emu.tests](#), 15
 - [emu.tests.common](#), 15
 - [emu.tests.test_wps_bbox](#), 15
 - [emu.tests.test_wps_caps](#), 15
 - [emu.tests.test_wps_chomsky](#), 16
 - [emu.tests.test_wps_dummy](#), 16
 - [emu.tests.test_wps_hello](#), 16
 - [emu.tests.test_wps_inout](#), 16
 - [emu.tests.test_wps_ultimate_question](#), 16
 - [emu.tests.test_wps_wordcounter](#), 16

B

BinaryOperator (class in emu.processes.wps_binaryoperator), 11
 BinaryOperator._handler() (in module emu.processes.wps_binaryoperator), 12
 Box (class in emu.processes.wps_bbox), 11
 Box._handler() (in module emu.processes.wps_bbox), 11

C

Chomsky (class in emu.processes.wps_chomsky), 12
 Chomsky._handler() (in module emu.processes.wps_chomsky), 12
 client_for() (in module emu.tests.common), 15

D

Dummy (class in emu.processes.wps_dummy), 12
 Dummy._handler() (in module emu.processes.wps_dummy), 12

E

emu (module), 11
 emu.processes (module), 11
 emu.processes.wps_bbox (module), 11
 emu.processes.wps_binaryoperator (module), 11
 emu.processes.wps_chomsky (module), 12
 emu.processes.wps_dummy (module), 12
 emu.processes.wps_error (module), 13
 emu.processes.wps_inout (module), 13
 emu.processes.wps_multiple_outputs (module), 13
 emu.processes.wps_nap (module), 14
 emu.processes.wps_say_hello (module), 14
 emu.processes.wps_sleep (module), 14
 emu.processes.wps_ultimate_question (module), 14
 emu.processes.wps_wordcounter (module), 15
 emu.tests (module), 15
 emu.tests.common (module), 15
 emu.tests.test_wps_bbox (module), 15
 emu.tests.test_wps_caps (module), 15
 emu.tests.test_wps_chomsky (module), 16

emu.tests.test_wps_dummy (module), 16
 emu.tests.test_wps_hello (module), 16
 emu.tests.test_wps_inout (module), 16
 emu.tests.test_wps_ultimate_question (module), 16
 emu.tests.test_wps_wordcounter (module), 16

I

InOut (class in emu.processes.wps_inout), 13
 InOut._handler() (in module emu.processes.wps_inout), 13

M

MultipleOutputs (class in emu.processes.wps_multiple_outputs), 13
 MultipleOutputs._handler() (in module emu.processes.wps_multiple_outputs), 13

N

Nap (class in emu.processes.wps_nap), 14
 Nap._handler() (in module emu.processes.wps_nap), 14

S

SayHello (class in emu.processes.wps_say_hello), 14
 SayHello._handler() (in module emu.processes.wps_say_hello), 14
 ShowError (class in emu.processes.wps_error), 13
 ShowError._handler() (in module emu.processes.wps_error), 13
 Sleep (class in emu.processes.wps_sleep), 14
 Sleep._handler() (in module emu.processes.wps_sleep), 14

T

test_wps_bbox() (in module emu.tests.test_wps_bbox), 15
 test_wps_caps() (in module emu.tests.test_wps_caps), 15
 test_wps_chomsky() (in module emu.tests.test_wps_chomsky), 16

`test_wps_dummy()` (in module `emu.tests.test_wps_dummy`), [16](#)
`test_wps_hello()` (in module `emu.tests.test_wps_hello`), [16](#)
`test_wps_inout()` (in module `emu.tests.test_wps_inout`), [16](#)
`test_wps_ultimate_question()` (in module `emu.tests.test_wps_ultimate_question`), [16](#)
`test_wps_wordcount()` (in module `emu.tests.test_wps_wordcounter`), [16](#)

U

`UltimateQuestion` (class in `emu.processes.wps_ultimate_question`), [14](#)
`UltimateQuestion._handler()` (in module `emu.processes.wps_ultimate_question`), [15](#)

W

`WordCounter` (class in `emu.processes.wps_wordcounter`), [15](#)
`WordCounter._handler()` (in module `emu.processes.wps_wordcounter`), [15](#)
`WpsTestClient` (class in `emu.tests.common`), [15](#)
`WpsTestClient.get()` (in module `emu.tests.common`), [15](#)