ROyWeb Documentation

Release 0.9.1

Tamas Gal

May 19, 2015

Contents

CHAPTER 1

Getting the latest version

The original git-repository is on GitHub: https://github.com/tamasgal/royweb

CHAPTER 2

Installing dependencies

ROyWeb is based on Python and needs two additional libraries: Tornado and daemon.

2.1 Setting up an independent Python environment via virtualenv

The easiest way to set up a working environment is to install virtualenv first (if not already installed). Grab the latest version via:

```
curl -O https://pypi.python.org/packages/source/v/virtualenv/virtualenv-X.X.tar.gz
tar xvfz virtualenv-X.X.tar.gz
cd virtualenv-X.X
python virtualenv.py royweb
```

This will install a freshly new configured python in its own directory within the virtualenv folder. From now on, you can simply activate the virtual environment via:

```
cd royweb
source bin/activate
```

and mess around with it. Btw. I recommend having a look at virtualenvwrapper, which is a great addition to the already awesome virtualenv tool.

2.2 Easier sandboxing via virtualenvwrapper

As I already wrote, I recommend `virtualenvwrapper` to manage your Python environments, which of course you can install via `pip`:

```
pip install virtualenvwrapper
```

And then follow the instructions to setup your virtualenvwrapper environment. It is very simple, you only have to put a line into your shell startup script (`.bashrc` or whatever) and define a working directory and another directory to store your virtual envorionments. After that, the workflow to create an independent Python environment is:

```
mkproject royweb
pip install royweb
```

To exit the environment simply type:

```
deactivate
```

And to switch back anytime, use the `workon` command:

```
workon royweb
```

This way you can create a bunch of virtual envrionemnts which are completely clean and independent from each other.

2.3 Installing the dependencies when using the source

If you want to work on the develeopment version, you'll need to install Tornado and daemon. This is easy as pip is:

```
pip install tornado
pip install daemon
```

You can also use the `requirements.txt` file to install all dependencies automatically:

```
pip install -r requirements.txt
```

From now on, every time you want to start a ROyWeb server, activate the virtual environment and you're ready to go.

CHAPTER 3

Using ROyWeb

3.1 Integrate ROyWeb into your project

There is a class `PacketHandler`, which can be used to create and send JSON UDP packets with the required format. If you want to monitor some values in your projects, initialise a `PacketHandler` and use its `send()` method to transfer the values. Here is an example:

```
from royweb import PacketHandler
ph = PacketHandler("127.0.0.1", 9999)
ph.send('foo', 23, 'This is a description')
```

That's it ;-)

3.2 JSON message format

The UDP packet which contains the data to plot later on should only include a stringified JSON dump in its UDP data field. The required JSON structure is:

```
{
    'kind': 'parameter',
    'type': 'parameter_short_name',
    'unit': 'desired_unit',
    'description': 'This is a longer description of the parameter.',
    'value': 'the_value'
}
```

The `kind` field is required and should always be `"parameter"` (ROyWeb uses the same JSON format for other "kinds" communications). The value can be of any type, but it will actually determine which types of graphs you can plot later on. For `TimePlots`, `Histograms` and `Equalisers`, simply use a `string` for example, it will be converted to float on the fly.

CHAPTER 4

API Documentation

4.1 Module *networking*

Networking stuff for UDP and WebSocket communication.

class `royweb.networking.`**`PacketHandler`**(*ip*, *port*)
A reusable packet handler, which can send parameters via UDP.

`json_message`(*parameter_type*, *value*, *unit*, *description*)
Create a json dump.

`send`(*parameter_type*, *value*, *unit*, *description*)
Send a parameter with value and description to a ROyWeb

class `royweb.networking.`**`WebSocketBroadcaster`**(*server*, *port*, *clients*)
Receives data from UDP and redistributes them via WebSockets.

`run`()
Listen for UDP packets and immediately send them to the clients.

`stop`()
Stop the port listener.

`with_timestamp`(*json_obj*)
Returns a copy of a json obj with an additional time property.

4.2 Module *webhandler*

Tornado WebHandler.

class `royweb.webhandler.`**`MainHandler`**(*application*, *request*, ***kwargs*)
The main request handler for ROyWeb

class `royweb.webhandler.`**`EchoWebSocket`**(**args*, ***kwargs*)
An echo handler for client/server messaging and debugging

`send_json_message`(*text*)
Convert message to json and send it to the clients

class `royweb.webhandler.`**`NoCacheStaticFileHandler`**(*application*, *request*, ***kwargs*)
A static file handler without caching.

class `royweb.webhandler.`**`UnitTests`**(*application*, *request*, ***kwargs*)
The unit test page

class royweb.webhandler.**SpecTests**(*application*, *request*, ***kwargs*)
The specs page

CHAPTER 5

royweb

Restless Oyster Web is an online monitoring tool. It provides a graphical user interface which can be accessed by any WebSocket-capable web browser. Although it's designed for the KM3NeT neutrino detector, it provides a simple interface to let you monitor other kinds of parameters. The current status is beta and is already used by many people. Thanks for you feedback so far!

CHAPTER 6

Future Plans

The main goal is to create a tool which runs a web interface and monitors several types of parameters sent via UDP packets. The visualisation of the data is done by the d3.js framework and the parameters are sent as JSON objects. Monitoring a specific parameter should be as easy as sending a JSON object through UDP to the webserver. Any connected client should then receive the packet and the user will be able to create live graphs or histograms with the desired parameters.

Feel free to contribute or join; any kind of feedback is welcome!

6.1 Documentation

Read the docs at http://royweb.readthedocs.org

6.2 Live Demo

A very basic demonstration can be seen at http://royweb.km3net.de Please note that the server is not always running the latest build.

6.3 Installation

I highly recommend using virtualenv for any Python related experiments.

After you set up a seperate virtual environment, use `pip` to install the latest release:

```
pip install royweb
```

This will automatically install all dependencies and scripts. Of course, you can also download the source and discover the code on your own.

6.4 Simple usage

If you installed `royweb` via `pip`, you can use the `royweb` script to start the web server with the default configuration. Otherwise, simply take the `royweb.py` in the `royweb` package. The server will listen to incoming client connections on port **8080** and start a UDP-listener on port **9999** for parameter monitoring:

```
# royweb
Starting ROyWeb with PID 25674
Running on 127.0.0.1:8080
Listening for UDP data on port 9999
```

6.5 Send test data

To send some live test data to the web server, run `royweb_tester` (if installed via `pip`) or the `send_udp.py` script. This will generate some random parameters and distributes them via UDP to the default port **9999** on localhost:

```
# royweb_tester
UDP target IP: 127.0.0.1
UDP target port: 9999
```

Open your browser and navigate to http://127.0.0.1:8080 to see the live parameter logging.

6.6 Integrate ROyWeb into your project

There is a class `PacketHandler`, which can be used to create and send JSON UDP packets with the required format. If you want to monitor some values in your projects, initialise a `PacketHandler` and use its `send()` method to transfer the values. Here is an example:

```
from royweb import PacketHandler
ph = PacketHandler("127.0.0.1", 9999)
ph.send('foo', 23, 'This is a description')
```

That's it ;-)

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*

r

E

J

M

N

P

R

S

U

W