
ipywidgets_server Documentation

Release 0.2.2-alpha2

Pascal Bugnion

Dec 10, 2018

Contents

1	Getting started: 30 seconds with IPywidgets server	3
2	Installation	5
3	Tutorials	7
3.1	Simple widget	7
4	Developing <i>ipywidgets-server</i>	11
4.1	Installation	11
4.2	Running in debug mode	11
5	Release notes	13
5.1	Version 0.2.1	13
5.2	Version 0.2.0	13

IPywidgets server is no longer maintained. To serve widgets and rich outputs, we recommend using [voila](#) instead. See [this issue](#) for a discussion.

IPywidgets server lets you serve Jupyter widgets outside of a Jupyter Notebook. Any Python callback defined on your widgets will work as in the notebook.

CHAPTER 1

Getting started: 30 seconds with IPywidgets server

Let's create a simple widget:

```
# example.py

from ipywidgets import IntSlider, Text, VBox
s = IntSlider(max=200, value=100)
t = Text()

def update_text(change=None):
    t.value = str(float(s.value) ** 2)

s.observe(update_text, names='value')
update_text()
vbox = VBox([s, t])
```

To serve this, just run the following, in the directory containing `example.py`:

```
$ ipywidgets-server example:vbox
```

This will serve the widget on `http://localhost:8866/`:

The command line argument, `example:vbox`, is `<module name>:<object name>`, where `<module name>` is the name of a Python module that can be imported (for instance, a file in the current directory, without the `.py` extension or a Python package that is installed in your environment), and `<object name>` is the name of the variable that holds the widget to display.

For information on other command line arguments, run:

```
$ ipywidgets-server --help
```


CHAPTER 2

Installation

Install *IPywidgets server* with:

```
$ pip install ipywidgets_server
```

Currently, *IPywidgets server* only runs on Python 3.6.

3.1 Simple widget

Let's build a stock price fetcher. We will allow the user to enter a stock ticker symbol and fetch information about the current stock price from the [Quandl API](#). The code for this example is also available in [examples/stock_price](#).



← → ↻ ⓘ 127.0.0.1:8866

GOOG FETCH

1177.33

```
# example.py

from urllib.request import urlopen
from urllib.error import HTTPError

import json

import ipywidgets as widgets

BASE_URL = 'https://www.quandl.com/api/v3/datasets/WIKI/{}.json?rows=1'

stock_input = widgets.Text('GOOG')
```

(continues on next page)

(continued from previous page)

```

fetch_button = widgets.Button(description='FETCH')
result_container = widgets.HBox([
    widgets.Text(disabled=True)
])

def get_stock_price(symbol):
    """ Fetch stock price for `symbol` from quandl API """
    with urlopen(BASE_URL.format(symbol)) as response:
        response_json = json.loads(response.read())
    return response_json['dataset']['data'][0][1]

def handle_fetch_error(error, symbol):
    if error.code == 404:
        message = 'Stock symbol {} not found'.format(symbol)
    else:
        message = 'Unexpected error'
    result_container.children = [widgets.Label(message)]

def on_button_click(arg):
    """ Callback when 'fetch' button is clicked """
    # Disable widgets to give feedback that something is happening
    stock_input.disabled = True
    fetch_button.disabled = True

    # Fetch the stock price
    symbol = stock_input.value.upper()
    try:
        stock_price = get_stock_price(symbol)
        result_container.children = [
            widgets.Text(str(stock_price), disabled=True)
        ]
    except HTTPError as error:
        handle_fetch_error(error, symbol)

    # Re-enable the widgets
    stock_input.disabled = False
    fetch_button.disabled = False

fetch_button.on_click(on_button_click)

container = widgets.VBox([
    widgets.HBox([stock_input, fetch_button]),
    result_container
])

```

To run the example, enter this shell command:

```
$ ipywidgets-server example:container
```

This will serve this example on the default port, 8866. Visit <http://127.0.0.1:8866> with your browser, and you should see the widget.

We note the following:

- we wrap all the widgets into a top-level container, container. This is the widget we then ask

`ipywidgets-server` to display.

- when the user presses the `FETCH` button, they should receive some feedback that the application is processing their request. Disabling the user inputs is an easy way to provide this feedback.
- depending on whether the stock price fetch was successful, we either display a text widget with the stock price, or a label widget with an error message. To render different widgets conditionally, we anchor a container (an `HBox`) and swap the container's children.

4.1 Installation

To develop *ipywidgets-server*, clone the [repository](#) using git. Then, go into the project root and run:

```
pip install -e .
```

This will build the JavaScript and install the Python in [editable mode](#).

If you make changes to the `js/` directory, you will need to rebuild the frontend:

```
cd js/  
npm run build
```

You will then need to refresh any open browser pages that contain *ipywidgets-server* instances. If you are making many changes to the `js/` directory, you can automatically build the frontend for every change with:

```
npm run build:watch
```

If you make changes to the Python side, you will need to restart running *ipywidgets-server* instances. If you make changes to the *static* directory, you just need to refresh the browser page.

4.2 Running in debug mode

You can run *ipywidget-server* in debug mode with:

```
ipywidgets-server example:widget --WidgetsServer.log_level=DEBUG
```

You may also want to increase the log-level of the kernel driver. For this, change the log level in `ipywidgets_server/kernel/__main__.py`.

5.1 Version 0.2.1

This patch release:

- Ensures kernels are shut down when the client closes gracefully (PR 22)
- Adds developer documentation (PR 24)
- Improves the stock price example (PR 25)

5.2 Version 0.2.0

This minor release:

- Changes the default port to 8866 to avoid clashing with the default notebook port (PR 21)
- Makes sure that errors caught in the widget callbacks get propagated to the user (PR 23)
- Adds documentation (PR 14)