

---

# **SPADE Bokeh Server Plugin Documentation**

*Release 0.1.2*

**Javi Palanca**

**Jun 13, 2019**



---

## Contents:

---

<b>1 SPADE Bokeh Server Plugin</b>	<b>1</b>
1.1 Features . . . . .	1
1.2 Credits . . . . .	1
<b>2 Installation</b>	<b>3</b>
2.1 Stable release . . . . .	3
2.2 From sources . . . . .	3
<b>3 Usage</b>	<b>5</b>
3.1 Agent setup . . . . .	6
3.2 Bokeh plot creation . . . . .	7
3.3 Web controller . . . . .	7
<b>4 spade_bokeh</b>	<b>9</b>
4.1 spade_bokeh package . . . . .	9
<b>5 Contributing</b>	<b>11</b>
5.1 Types of Contributions . . . . .	11
5.2 Get Started! . . . . .	12
5.3 Pull Request Guidelines . . . . .	13
5.4 Tips . . . . .	13
5.5 Deploying . . . . .	13
<b>6 Credits</b>	<b>15</b>
6.1 Development Lead . . . . .	15
6.2 Contributors . . . . .	15
<b>7 History</b>	<b>17</b>
7.1 0.1.2 (2018-09-17) . . . . .	17
7.2 0.1.0 (2018-09-17) . . . . .	17
<b>8 Indices and tables</b>	<b>19</b>
<b>Python Module Index</b>	<b>21</b>
<b>Index</b>	<b>23</b>



---

## SPADE Bokeh Server Plugin

---

Bokeh server Plugin for SPADE agents that integrates with agents web service. Allows to render dynamic bokeh plots inside the agent templates in a very easy way.

- Free software: MIT license
- Documentation: <https://spade-bokeh.readthedocs.io>.

### 1.1 Features

- TODO

### 1.2 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.



## 2.1 Stable release

To install SPADE Bokeh Server, run this command in your terminal:

```
$ pip install spade_bokeh
```

This is the preferred method to install SPADE Bokeh Server, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

## 2.2 From sources

The sources for SPADE Bokeh Server can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/javipalanca/spade_bokeh
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/javipalanca/spade_bokeh/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```





---

## Usage

---

**Note:** This is a plugin for the [SPADE](#) agent platform. Please visit the [SPADE's documentation](#) to know more about this platform.

---

Plotting figures in html usually involves too much javascript code. To solve this there is a great python library called [Bokeh](#) that helps us to plot our own figures with python code and dynamically thanks to a server embedded with bokeh.

This plugin provides a **mixin** to include a bokeh server in an agent and to register plots to be then rendered in your jinja2 templates of your agent's web interface.

To use SPADE Bokeh Server in a project:

```
from spade import agent
import spade_bokeh

from bokeh.layouts import column
from bokeh.models import ColumnDataSource, Slider
from bokeh.plotting import figure
from bokeh.sampledata.sea_surface_temperature import sea_surface_temperature

class MyBokehAgent(spade_bokeh.BokehServerMixin, agent.Agent):

    # agent setup
    async def setup(self):

        self.web.add_get("/plot", self.controller, "plot.html")

        self.web.start(port=10000)
        self.bokeh_server.start(hostname="localhost", port=5006)

        self.bokeh_server.add_plot("/my_plot", self.modify_doc)

    # bokeh plot creation
```

(continues on next page)

(continued from previous page)

```

def modify_doc(self, doc):
    """ here you create your plot, see the Bokeh documentation """
    df = sea_surface_temperature.copy()
    source = ColumnDataSource(data=df)

    plot = figure(x_axis_type='datetime', y_range=(0, 25),
                  y_axis_label='Temperature (Celsius)',
                  title="Sea Surface Temperature at 43.18, -70.43")
    plot.line('time', 'temperature', source=source)

    def callback(attr, old, new):
        if new == 0:
            data = df
        else:
            data = df.rolling('{0}D'.format(new)).mean()
            source.data = ColumnDataSource(data=data).data

    slider = Slider(start=0, end=30, value=0, step=1, title="Smoothing by N Days")
    slider.on_change('value', callback)

    doc.add_root(column(slider, plot))

# web controller
async def controller(self, request):
    script = self.bokeh_server.get_plot_script("/my_plot")

    return {"script": script}

```

In the example below there are 3 different blocks: the agent setup, the web controller and the bokeh plot creation.

## 3.1 Agent setup

To activate the *spade\_bokeh* plugin in your agent you must follow some steps:

- First of all your agent needs to inherit from *spade\_bokeh.BokehServerMixin*.

**Warning:** All the mixins MUST be included **before** the *spade.agent.Agent* class, to respect the method resolution order (e.g. `class MyAgent(MyMixin1, MyMixin2, ..., Agent)`).

- Next you need to start your bokeh\_server with the following order: `self.bokeh_server.start()`. This method may accept two arguments: the hostname and the port of the bokeh\_server. By default they are "localhost" and 5006.

**Warning:** Two agents can not share a same bokeh\_server, so they must use different ports!

- Finally, you can add new plots to your server using the `add_plot` function of bokeh\_server. It accepts two arguments: the *path* for the plot in the bokeh server and the *callback* to build the plot.

---

**Note:** You can add as many plots as you need.

---

## 3.2 Bokeh plot creation

The method that you give to the `add_plot` call is the callback that will create the plot when the url path is queried. Inside this method you can create your plot following the *Bokeh* guidelines. As in the example, the method receives a `doc` argument, where the plots of your application will be rendered.

---

**Hint:** To learn more about how to create *Bokeh* plots please visit the [Bokeh User Guide](#)

---

## 3.3 Web controller

The final step to render your bokeh plots inside an agent view is to render the plot in a template managed by the SPADE's web interface system.

---

**Hint:** Please, visit the SPADE's documentation to know more about how to create a SPADE web interface for your agents.

---

The *spade\_bokeh* plugin provides you a helper function to easily render your plots inside a jinja2 template. As in the example below, you can use the `get_plot_script` method with the path of the plot you want to render and it will return you the necessary javascript to render the plot (this javascript contains the URL and necessary code to connect to the bokeh server dinamycally).

Then you only need to render that script in your template as in the example:

```
<html lang="en">
  <head>
    <title>Bokeh Example</title>
  </head>

  <body>
    {{ script | safe}}
  </body>
</html>
```

---

**Note:** Note that you must *safe escape* the script with the `safe` jinja2 filter to avoid escaping the html tags.

---



## 4.1 spade\_bokeh package

### 4.1.1 Submodules

### 4.1.2 spade\_bokeh.spade\_bokeh module

**class** spade\_bokeh.spade\_bokeh.**BokehServerMixin** (\*args, \*\*kwargs)

Bases: object

This is the Mixin to inherit from when you create your agent.

**class** spade\_bokeh.spade\_bokeh.**BokehServer** (agent)

Bases: object

**add\_plot** (path, func)

Registers a new plot in the bokeh server. Args:

path: path where the plot will respond to queries func: the function that renders the plot.

**bokeh\_worker** ()

**get\_plot\_script** (path)

Returns the necessary javascript to render a plot Args:

path (str): the path with which the plot was registered in the server.

**Returns:** A string with the javascript code to render the plot.

**start** (hostname='localhost', port=5006)

Starts the bokeh server. Args:

hostname (str): hostname of the server. Must be the same where the agent is running. Defaults to "localhost" port (int): port of the server. Defaults to 5006.

**stop ()**  
Stops the Bokeh server.

### 4.1.3 Module contents

Top-level package for SPADE Bokeh Server.

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 5.1 Types of Contributions

### 5.1.1 Report Bugs

Report bugs at [https://github.com/javipalanca/spade\\_bokeh/issues](https://github.com/javipalanca/spade_bokeh/issues).

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 5.1.4 Write Documentation

SPADE Bokeh Server could always use more documentation, whether as part of the official SPADE Bokeh Server docs, in docstrings, or even on the web in blog posts, articles, and such.

## 5.1.5 Submit Feedback

The best way to send feedback is to file an issue at [https://github.com/javipalanca/spade\\_bokeh/issues](https://github.com/javipalanca/spade_bokeh/issues).

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 5.2 Get Started!

Ready to contribute? Here's how to set up *spade\_bokeh* for local development.

1. Fork the *spade\_bokeh* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/spade_bokeh.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv spade_bokeh
$ cd spade_bokeh/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 spade_bokeh tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.



## 5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check [https://travis-ci.org/javipalanca/spade\\_bokeh/pull\\_requests](https://travis-ci.org/javipalanca/spade_bokeh/pull_requests) and make sure that the tests pass for all supported Python versions.

## 5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_spade_bokeh
```

## 5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.



## 6.1 Development Lead

- Javi Palanca <jpalanca@gmail.com>

## 6.2 Contributors

None yet. Why not be the first?



### 7.1 0.1.2 (2018-09-17)

- Added requirements file.
- Fixed tox configuration for CI.
- Fixed readthedocs configuration.

### 7.2 0.1.0 (2018-09-17)

- First release on PyPI.



## CHAPTER 8

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





**S**

[spade\\_bokeh](#), 10

[spade\\_bokeh.spade\\_bokeh](#), 9



## A

`add_plot()` (*spade\_bokeh.spade\_bokeh.BokekServer*  
*method*), 9

## B

`bokeh_worker()` (*spade\_bokeh.spade\_bokeh.BokekServer*  
*method*), 9

`BokehServerMixin` (*class* *in*  
*spade\_bokeh.spade\_bokeh*), 9

`BokekServer` (*class in spade\_bokeh.spade\_bokeh*), 9

## G

`get_plot_script()`  
(*spade\_bokeh.spade\_bokeh.BokekServer*  
*method*), 9

## S

`spade_bokeh` (*module*), 10

`spade_bokeh.spade_bokeh` (*module*), 9

`start()` (*spade\_bokeh.spade\_bokeh.BokekServer*  
*method*), 9

`stop()` (*spade\_bokeh.spade\_bokeh.BokekServer*  
*method*), 9