voila

Release 0.1.8

The Voila Development Team

Jul 23, 2019

CONTENTS

From notebooks to standalone web applications and dashboards.

Voilà allows you to convert a Jupyter Notebook into an interactive dashboard that allows you to share your work with others. It is secure and customizable, giving you control over what your readers experience.

For example, here's a dashboard created with Voila. (you can try it interactively at the following Binder link)

For more information about Voila, see the sections below.

CHAPTER

ONE

INSTALLING VOILA

Voila can be installed with the `conda` package manager

```
conda install -c conda-forge voila
```

or from `PyPI`:

```
pip install voila
```

Once Voila is installed, it can be used either as a Command-Line Interface, or as a Jupyter Server extension. See *Using Voila* for information on how to use Voila.

CHAPTER

TWO

USING VOILA

Voila can be used as a standalone application, or as a Jupyter server extension. This page describes how to do each. Before you begin, make sure that you follow the steps in *Installing Voila*.

The following sections cover how to use Voila.

2.1 As a standalone application

Voila can be used to run, convert, and serve a Jupyter notebook as a standalone app. This can be done via the command-line, with the following pattern:

```
voila <path-to-notebook> <options>
```

For example, to render the `bqplot` example notebook as a standalone app, run

```
git clone https://github.com/QuantStack/voila
cd voila
voila notebooks/bqplot.ipynb
```

Voila display a message when your notebook-based application is live. By default, Voila runs at `localhost:8866`.

To serve a **directory of Jupyter Notebooks**, navigate to the directory you'd like to serve, then simply run `voila`:

```
cd notebooks/
voila
```

The page served by Voila will now contain a list of any notebooks in the directory. By clicking on one, you will trigger Voila's conversion process. A new Jupyter kernel will be created for each notebook you click.

2.2 As a Jupyter server extension

You can also use Voila from within a Jupyter server (e.g., after running `jupyter lab` or `jupyter notebook`).

Note: Voila can also be used as a notebook server extension, both with the notebook server or with the jupyter_server.

To use Voila within a pre-existing Jupyter server, first start the server, then go to the following URL:

```
<url-of-my-server>/voila
```

For example, if you typed `jupyter lab` and it was running at `http://localhost:8888/lab`, then Voila would be accessed at `http://localhost:8888/voila`.

In this case, Voila will serve the directory in which the Jupyter server was started.

2.3 How does Voila work?

When Voila is run on a notebook, the following steps occur:

1. Voila runs the code in the notebook and collects the outputs
2. The notebook and its outputs are converted to HTML. By default, the notebook **code cells are hidden**.
3. This page is served either as a Tornado application, or via the Jupyter server.
4. When users access the page, the widgets on the page have access to the underlying Jupyter kernel.

CHAPTER

THREE

CUSTOMIZING VOILA

There are many ways you can customize Voila to control the look and feel of the dashboards you create.

3.1 Controlling the nbconvert template

Voila uses **nbconvert** to convert your Jupyter Notebook into an HTML dashboard. nbconvert has a rich templating system that allows you to customize the way in which your Jupyter Notebook is converted into HTML.

By default, Voila will render the HTML from your notebook in the same linear fashion that the notebook follows. If you'd like to use a different layout, this can be controlled by creating a new nbconvert template, registering it with Voila, and calling it from the command-line like so:

```
voila <path-to-notebook> --template=<name-of-template>
```

For example, Voila includes one other template that uses a Javascript library and an alternate `<div>` layout in order to let the user drag and drop cells.

For example, to use the gridstack template, use the command:

```
voila <path-to-notebook> --template=gridstack
```

3.2 Creating your own template

You can create your own nbconvert template for use with Voila. This allows you to control the look and feel of your dashboard.

In order to create your own template, first familiarize yourself with **Jinja**, **HTML**, and **CSS**. Each of these is used in creating custom templates. For more information, see the nbconvert templates documentation. For one example, check out the nbconvert basic HTML template.

3.2.1 Where are Voila templates located?

All voila templates are stored as folders with particular configuration/template files inside. These folders can exist in the standard Jupyter configuration locations, in a folder called `voila/templates`. For example:

```
~/.local/share/jupyter/voila/templates
~/path/to/env/dev/share/jupyter/voila/templates
/usr/local/share/jupyter/voila/templates
/usr/share/jupyter/voila/templates
```

Voila will search these locations for a folder, one per template, where the folder name defines the template name.

3.2.2 The Voila template structure

Within each template folder, you can provide your own nbconvert templates, static files, and HTML templates (for pages such as a 404 error). For example, here is the folder structure of the base Voila template (called "default"):

```
tree path/to/env/share/jupyter/voila/templates/default/
├── nbconvert_templates
│   ├── base.tpl
│   └── voila.tpl
└── templates
    ├── 404.html
    ├── error.html
    ├── page.html
    └── tree.html
```

To customize the nbconvert template, store it in a folder called `templatename/nbconvert_templates/voila.tpl`. In the case of the default template, we also provide a `base.tpl` that our custom template uses as a base. The name `voila.tpl` is special - you cannot name your custom nbconvert something else.

To customize the HTML page templates, store them in a folder called `templatename/templates/<name>.html`. These are files that Voila can serve as standalone HTML (for example, the `tree.html` template defines how folders/files are displayed in `localhost:8866/voila/tree`). You can override the defaults by providing your own HTML files of the same name.

To configure your Voila template, you should add a `config.json` file to the root of your template folder.

3.2.3 An example custom template

To show how to create your own custom template, let's create our own nbconvert template. We'll have two goals:

1. Add an `<h1>` header displaying "Our awesome template" to the voila dashboard.
2. Add a custom 404.html page that displays an image.

First, we'll create a folder in `~/.local/share/jupyter/voila/templates` called `mytemplate`:

```
mkdir ~/.local/share/jupyter/voila/templates/mytemplate
cd ~/.local/share/jupyter/voila/templates/mytemplate
```

Next, we'll copy over the base template files for voila, which we'll modify:

```
cp -r path/to/env/share/jupyter/voila/templates/default/nbconvert_templates ./
cp -r path/to/env/share/jupyter/voila/templates/default/templates ./
```

We should now have a folder structure like this:

```
tree .
├── nbconvert_templates
│   ├── base.tpl
│   └── voila.tpl
└── templates
    ├── 404.html
    ├── error.html
    ├── page.html
    └── tree.html
```

Now, we'll edit `nbconvert_templates/voila.tpl` to include a custom H1 header.

As well as `templates/tree.html` to include an image.

Finally, we can tell `Voila` to use this custom template the next time we use it on a Jupyter notebook by using the name of the folder in the `--template` parameter:

```
voila mynotebook.ipynb --template=mytemplate
```

The result should be a Voila dashboard with your custom modifications made!

3.3 Adding your own static files

If you create your own theme, you may also want to define and use your own static files, such as CSS and Javascript. To use your own static files, follow these steps:

1. Create a folder along with your template (e.g., `mytemplate/static/`).
2. Put your static files in this template.
3. In your template file (e.g. `voila.tpl`), link these static files with the following path:

   ```
   {{resources.base_url}}voila/static/<path-to-static-files>
   ```

4. When you call `voila`, configure the static folder by using the `--static` kwarg, or by configuring `--VoilaConfiguration.static_root`.

Any folders / files that are inside the folder given with this configuration will be copied to `{{resources.base_url}}voila/static/`.

For example, if you had a CSS file called `custom.css` in `static/css`, you would link it in your template like so:

```
<link rel="stylesheet" type="text/css" href="{{resources.base_url}}voila/static/css/
↪custom.css"></link>
```

3.4 Configure voila for the Jupyter Server

Several pieces of `voila`'s functionality can be controlled when it is run. This can be done either as a part of the standalone CLI, or with the Jupyter Server. To configure `voila` when run by the Jupyter Server, use the following pattern when invoking the command that runs Jupyter (e.g., Jupyter Lab or Jupyter Notebook):

```
<jupyter-command> --VoilaConfiguration.<config-key>=<config-value>
```

For example, to control the template used by `voila` from within a Jupyter Lab session, use the following command when starting the server:

```
jupyter lab --VoilaConfiguration.template=distill
```

When users run `voila` by hitting the `voila/` endpoint, this configuration will be used.