Nbconvert as a webservice (rendering ipynb to static HTML) http://nbviewer.ipython.org

rgbkrk Merge pull request #572 from minrk/cache-upstream ··· Latest commit f282e1a 22 hours ago

nbviewer	don't re-use rate limit headers on cached responses	a day ago
.dockerignore	updating .ignores	10 months ago
.gitignore	updating .ignores	10 months ago
.travis.yml	use jupyter/notebook for docker	6 months ago
Dockerfile	install nbformat from master in Dockerfile	27 days ago
LICENSE.txt	relicensing	3 years ago
README.md	add missing `npm install` to README steps	3 months ago
docker-compose.yml	using python3: broken unicode redirect	10 months ago
package.json	more rebranding, some typos	4 months ago
requirements-dev.txt	use jupyter/notebook for docker	6 months ago
requirements.txt	more docker cleanup	6 months ago
setup.py	fixing #448: adding providers to pkg_data	10 months ago
tasks.py	Fixed imports of deprecated packages related to IPython.	6 months ago

README.md

Jupyter Notebook Viewer

gitter join chat

Jupyter nbviewer is the web application behind The Jupyter Notebook Viewer, which is graciously hosted by Rackspace.

Run this locally to get most of the features of nbviewer on your own network.

Quick Run

If you have `docker` installed, you can pull and run the currently built version of the Docker container by

```
$ docker pull jupyter/nbviewer
$ docker run -p 8080:8080 jupyter/nbviewer
```

It automatically gets built with each push to `master`, so you'll always be able to get the freshest copy.

For speed and friendliness to GitHub, be sure to set `GITHUB_OAUTH_KEY` and `GITHUB_OAUTH_SECRET`:

```
$ docker run -p 8080:8080 -e 'GITHUB_OAUTH_KEY=YOURKEY' \
                          -e 'GITHUB_OAUTH_SECRET=YOURSECRET' \
                          jupyter/nbviewer
```

Or to use your GitHub personal access token, you can set just `GITHUB_API_TOKEN`.

GitHub Enterprise

To use nbviewer on against your own GitHub Enterprise instance you need to set `GITHUB_API_URL`. The relevant API endpoints for GitHub Enterprise are prefixed with `http://hostname/api/v3`. You must also specify your `OAUTH` or `API_TOKEN` as explained above. For example:

```
$ docker run -p 8080:8080 -e 'GITHUB_OAUTH_KEY=YOURKEY' \
                          -e 'GITHUB_OAUTH_SECRET=YOURSECRET' \
                          -e 'GITHUB_API_URL=https://ghe.example.com/api/v3/' \
                          jupyter/nbviewer
```

With this configured all GitHub API requests will go to your Enterprise instance so you can view all of your internal notebooks.

Local Development

With Docker

You can build a docker image that uses your local branch.

Build

```
$ cd <path to repo>
$ docker build -t nbviewer .
```

Run

```
$ cd <path to repo>
$ docker run -p 8080:8080 nbviewer
```

With Docker Compose

The Notebook Viewer uses `memcached` in production. To locally try out this setup, a docker-compose configuration is provided to easily start/stop the `nbviewer` and `memcached` containers together from a your current branch. You will need to install `docker` prior to this.

Run

```
$ cd <path to repo>
$ pip install docker-compose
$ docker-compose up
```

Local Installation

The Notebook Viewer requires several binary packages to be installed on your system. The primary ones are `libmemcached-dev libcurl4-openssl-dev pandoc libevent-dev`. Package names may differ on your system, see salt-states for more details.

If they are installed, you can install the required Python packages via pip.

```
$ cd <path to repo>
$ pip install -r requirements.txt
```

Static Assets

Static assets are maintained with `bower` and `less` (which require having `npm` installed), and the `invoke` python module.

```
$ cd <path to repo>
$ pip install -r requirements-dev.txt
$ npm install
```

```
$ invoke bower
$ invoke less [-d]
```

This will download the relevant assets into `nbviewer/static/components` and create the built assets in `nbviewer/static/build`.

Pass `-d` or `--debug` to `invoke less` to create a CSS sourcemap, useful for debugging.

Running Locally

```
$ cd <path to repo>
$ python -m nbviewer --debug --no-cache
```

This will automatically relaunch the server if a change is detected on a python file, and not cache any results. You can then just do the modifications you like to the source code and/or the templates then refresh the pages.

Running the Tests

`nose` is used to run the test suite. The tests currently make calls to external APIs such as GitHub, so it is best to use your Github API Token when running:

```
$ cd <path to repo>
$ pip install -r requirements-dev.txt
$ GITHUB_API_TOKEN=<your token> python setup.py test
```

Extending the Notebook Viewer

Providers

Providers are sources of notebooks and directories of notebooks and directories.

`nbviewer` ships with several providers

- `url`
- `gist`
- `github`
- `local`

Writing a new Provider

There are several already additional providers proposed/requested. Some providers are more involved than others, and some, such as those which would require user authentication, will take some work to support properly.

A provider is implemented as a python module, which can expose a few functions:

`uri_rewrites`

If you just need to rewrite URLs (or URIs) of another site/namespace, implement `uri_rewrites`, which will allow the front page to transform an arbitrary string (usually an URI fragment), escape it correctly, and turn it into a "canonical" nbviewer URL. See the dropbox provider for a simple example of rewriting URLs without using a custom API client.

`default_handlers`

If you need custom logic, such as connecting to an API, implement `default_handlers`. See the github provider for a complex example of providing multiple handlers.

Error Handling

While you *could* re-implement upstream HTTP error handling, a small convenience method is provided for intercepting HTTP errors. On a given URL handler that inherits from `BaseHandler`, overload the `client_error_message` and re-call it with your message (or `None`). See the gist provider for an example of customizing the error message.

Formats

Formats are ways to present notebooks to the user.

`nbviewer` ships with two providers:

- `html`
- `slides`

Writing a new Format

If you'd like to write a new format, open a ticket, or speak up on gitter! We have some work yet to do to support your next big thing in notebook publishing, and we'd love to here from you.