export_highlights

October 21, 2015

1 Exporting the notebook

As suggested by @juhasch, it is interesting to keep the highlights when exporting the notebook to another format. We give and explain below some possibilities:

1.1 Short version

- Html export:

  ```
  jupyter nbconvert FILE  --config JUPYTER_DATA_DIR/extensions/highlight_html_cfg.py
  ```

- LaTeX export:

  ```
  jupyter nbconvert FILE  --config JUPYTER_DATA_DIR/extensions/highlight_latex_cfg.py
  ```

  where JUPYTER_DATA_DIR can be found from the output of

  ```
  jupyter --paths
  ```

  eg `~/.local/share/jupyter` in my case. Seems to be `c:\users\NAME\AppData\Roaming\jupyter` under Windows.

Examples can be found here: initial notebook, html version, pdf version (after an additional LaTeX → pdf compilation).

1.2 Html export

This is quite easy. Actually, highlight formatting embedded in markdown cells is preserved while converting with the standard

```
jupyter nbconvert file.ipynb
```

However, the css file is missing and must be added. Here we have several possibilities

- Embed the css *within* the notebook. For that, consider the last cell of the present notebook. This code reads the css file `highlighter.css` in the extension directory and displays the corresponding style. So doing the <style> ...</style> section will be present in the cell output and interpreted by the web browser. Drawbacks of this solution is that user still have to execute this cell and that the this is not language agnostic.
- Use a **template file** to link or include the css file during conversion. Such a file is provided as `templates/highlighter.tpl`. It was choosen here to *include* the css content in the produced html file rather than linking it. This avoids the necessity to keep the css file with the html files.
- This works directly if the css resides in the same directory as the file the user is attempting to convert –thus requires the user to copy `highlighter.css` in the current directory. Then the conversion is simply

```
jupyter nbconvert file.ipynb --template highlighter
```

- It still remains two problems with this approach. First, it can be annoying to have to systematically copy the css file in the current directory. Second, the data within the html tags is not converted (and thus markdown remains unmodified). A solution is to use a pair of preprocessor/postprocessor that modify the html tags and enable the subsequent markdown to html converter to operate on the included data. Also, a config file is provided which redefines the template path to enable direct inclusion of the css file in the extension directory. Unfortunately, it seems that the *full path* to the config file has to be provided. This file resides in the extensions subdirectory of the jupyter_data_dir. The path can be found by looking at the output of

  ```
  jupyter --paths
  ```

  Then the command to issue for converting the notebook to html is

  ```
  jupyter nbconvert FILE  --config JUPYTER_DATA_DIR/extensions/highlight_html_cfg.py
  ```

For instance

```
jupyter nbconvert tst_highlights.ipynb  --config ~/.local/share/jupyter/extensions/highlight_html_cfg.py
```

1.3 LaTeX export

This is a bit more complicated since the direct conversion removes all html formatting present in markdown cells. Thus use again a **preprocessor** which runs before the markdown → LaTeX conversion. In turn, it appears that we also need to postprocess the result.

Three LaTeX commands, namely *highlighta, highlightb, highlightc*, and three environments *highlightA, highlightB, highlightC* are defined. Highlighting html markup is then transformed into the corresponding LaTeX commands and the text for completely highlighted cells is put in the adequate LaTeX environment.

Pre and PostProcessor classes are defined in the file `pp_highlighter.py` located in the `extensions` directory. A LaTeX template, that includes the necessary packages and the definitions of commands/environments is provides as `highlighter.tplx` in the template directory. The template inherits from `article.ltx`. For more complex scenarios, typically if the latex template file has be customized, the user shall modify its template or inherit from his base template rather than from article.

Finally, a config file fixes the different options for the conversion. Then the command to issue is simply

```
jupyter nbconvert FILE  --config JUPYTER_DATA_DIR/extensions/highlight_latex_cfg.py
```

e.g.

```
jupyter nbconvert tst_highlights.ipynb  --config ~/.local/share/jupyter/extensions/highlight_latex_cfg.p
```

1.4 Configuring paths

For those who do not have taken the extension from the `IPython-notebook-extensions` repository or have not configured extensions via its `setup.py` utility, a file `set_paths.py` is present in the extension directory (it is merely a verbatim copy of the relevant parts in setup.py). This file configure the paths to the `templates` and `extension` directories. It should be executed by something like

```
python3 set_paths.py
```

Additionaly, you may also have to execute `mv_paths.py` if you installed from the original repo via `jupyter nbextension install ..`

```
python3 mv_paths.py
```

1.5 Example for embedding the css within the notebook before conversion

```
>>> from IPython.core.display import display, HTML
... from jupyter_core.paths import jupyter_config_dir, jupyter_data_dir
... import os
... csspath=os.path.join(jupyter_data_dir(),'nbextensions','usability',
...                          'highlighter','highlighter.css')
... HTML('<style>'+open(csspath, "r").read()+'</style>')
```

```
<IPython.core.display.HTML object>
```