REPORT ON OpenDreamKit DELIVERABLE D2.7

Community-curated indexing tool (open source)

LUCA DE FEO

Due on	31/08/2017 (M24)
Delivered on	31/08/2018
Lead	Université de Versailles Saint-Quentin (UVSQ)
Progress on and finalization of this deliverable has been tracked publicly at: `https://github.com/OpenDreamKit/OpenDreamKit/issues/47`	

1. Introduction

The objective of task T2.10 is to deliver an open-source community-curated indexing tool for resources in computational mathematics. The goal is to collect examples, tutorials, lessons, and exercises related to a system or a field, under a unique hub, while maintaining the quality of the content through community curation.

Such a need has been felt in all communities involved in OpenDreamKit, and each has come up with its own solutions. The joint development efforts under OpenDreamKit, and in particular the unifying force of the common Jupyter notebook infrastructure, have given a unique occasion to produce a unified solution applicable to all systems. This report introduces *planetaryum*[1], a web toolkit for building curated Jupyter notebook collections.

Following the OpenDreamKit philosophy, *planetaryum* is not a single piece of software, but rather a toolkit, meant to power many different flavors of curated collections. While it may not cover all possible use cases, it is versatile enough to adapt to many needs of the community.

2. History

The need for maintaining various types of community curated help resources has long been felt in any community involved in OpenDreamKit. Some interesting examples are:

- Most computer algebra systems maintain a section of their website[2][3] containing links to high quality resources on the web.
- Some systems also have a wiki, e.g. `http://wiki.sagemath.org/`.
- The French SageMath community also used to host a well curated wiki with pointers to many didactic resources. The wiki was taken down due to maintenance difficulties.
- SageMath maintains its own Q&A website at `https://ask.sagemath.org`, whereas most other systems rely on existing generic solutions such as StackOverflow[4].
- For a long time, SageMath had been hosting a public instance of a SageMath server at `http://sagenb.org/`, where anyone could publish SageMath notebooks (old format incompatible with Jupyter) for everyone to view. The server had to be taken down among maintenance and security issues.
- The Jupyter project has a wiki page with a list of hand-selected notebooks[5].

[1] `https://github.com/OpenDreamKit/planetaryum`.
[2] `http://www.gap-system.org/Doc/Examples/examples.html`.
[3] `http://www.sagemath.org/help.html`.
[4] `https://stackoverflow.com/`.
[5] `https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks`.

More recently, the Jupyter community has provided the NBViewer service[6]. It is a static notebook previewer service that takes as input a URL pointing to a `.ipynb` file, and renders a static version of it. The Binder service[7] further lets the user edit and run a copy of the notebook. Importantly, those services do not host notebooks; they only render them (possibly temporarily caching the rendered result).

Following the success of the Jupyter format, major code hosting services like GitHub, GitLab, or BitBucket have started rendering static versions of Jupyter notebooks without relying on NBViewer.

The availability of these services has spurred a proliferation of collections of notebooks hosted on them, presented to the public through either NBViewer or the service-builtin preview functionality. However, this practice has the major inconvenience of making it hard to search, classify, and rank the notebooks.

When we sat down to plan for this deliverable, we wanted to provide a solution to host, search, and rank public resources, produced and curated by the community itself.

We started by evaluating available solutions. The first that we studied was Géant OER[8], a metadata aggregator for multimedia content. It quickly became apparent that its focus on metadata and multimedia did not suit our needs.

We also evaluated the *nbgallery*[9] software, a solution for hosting, indexing, searching, and ranking Jupyter notebooks. Despite its potential, the application is rather unstable, and support is limited given that it is essentially an internal project maintained by one person. Indeed, we didn't manage to install a fully working instance. Plus, although its advanced features are quite impressive, it does not cover all the use cases we were interested in.

Only two months ago, *QuantEcon Notes*[10] a curated collection of Jupyer notebooks for economic modelling, was put online by the QuantEcon project[11]. We reached the QuantEcon coordinators to ask about the software powering their platform, and learned that they plan to open source it under the name "Bookshelf". The Bookshelf project has an important overlap with the *planetaryum* project presented here; however, having learned so late about it, we were not able to benefit from it. We plan to look for synergies in the future, as we learn more on the scope and the features of the Bookshelf project.

Finally, we experimented with a custom developed application[12], whose development had already started in 2015 at a SageMath meeting. The application is capable of indexing, mirroring, searching and ranking all types of resources found on the web, with dedicated treatment for the most relevant formats, such as PDF, Jupyter, SageMath, HTML, etc. Unfortunately, when an alpha version of the service was deployed, it soon became apparent that it was not being adopted by the community at large. This was partially due to the service not being sufficiently easy to discover and use. But in the end, it simply looked like the community did not feel the need for such a generic tool, that was essentially trying (poorly) to replicate the job of a web search engine.

3. PLANETARYUM

After our first failed attempts, planetaryum arose as a new take on the problem. We realized that it was essential to recenter our efforts on a well-defined type of document, rather than

[6] `http://nbviewer.jupyter.org/`.
[7] `https://mybinder.org`.
[8] `https://oer.geant.org/`.
[9] `https://github.com/nbgallery/nbgallery`.
[10] `http://notes.quantecon.org/`.
[11] `https://quantecon.org/`.
[12] `http://sageindex.lipn.univ-paris13.fr/`.

dispersing our users in a format agnostic aggregator such as the sageindex experiment[13]. The generalization of the Jupyter notebook as a common document format for all OpenDreamKit systems presented a unique occasion to host instructional resources in a unified way.

Planetaryum also came with the realization that not a single application can fill all the user needs. We wanted to cover the teacher hosting a gallery of a dozen notebooks on their course web page, as well as the software community hub hosting thousands of user-contributed notebooks.

3.1. Use cases

Planetaryum is a modular Python library that can be used to build many different applications with a few lines of code. Some of the most requested applications are already bundled in the library and shipped as a command-line executable.

Here we present a few possible use cases for Planetaryum.

(1) **Static collection.** A (small) collection of notebooks can be used to generate a static website, based exclusively on HTML and JavaScript, and thus requiring very few resources for hosting. The appearance of the collection is customizable.

Generating a gallery is as simple as running one command:

```
planetaryum static -i notebook_dir -o output_dir
```

The website is generated in `output_dir`, and can be transferred to the hosting server by the usual means (e.g., FTP, ssh, etc.).

This application is similar to a statically generated website, such as one may produce using popular tools like Sphinx[14] or Jekyll[15], and indeed several examples of such deployments exist. However, having a dedicated tool adds specific features for Jupyter notebooks, such as keyword search, tailored layouts, etc. Another advantage is that it is easy to scale up to more complex applications as described below.

(2) **Medium sized collection, contributions via pull requests.** This model is suited for small to medium collections of notebooks where it is expected that the submission flow will be low and reserved to power users. It has the same advantages as the static collection, but at the same time it allows contributions, and can optionally be paired with a full-text search engine for better exploration.

The website generation can be automatized through continuous integration tools, as documented in the planetaryum sources[16][17]: from a Binder-ready GitHub repository containing notebooks it is possible to automatically generate and host on GitHub Pages a static view of the repository, any time its contents change, thanks to Travis CI.

We have set up a demonstration of this workflow at `https://opendreamkit.org/planetaryum-example-static/`.

(3) **Large collection, user uploads.** This is a full-fledged application, backed by a database and a full-text search engine. It features filtering, user voting, and potentially other advanced features such as recommendation. All the build and deploy steps are controlled from the planetaryum executable. It is very similar in spirit to *nbgallery*[18] or QuantEcon Notes[19], but it is built with the same components as the other applications.

[13] `http://sageindex.lipn.univ-paris13.fr/`.

[14] `http://www.sphinx-doc.org/`.

[15] `https://jekyllrb.com/`.

[16] `https://github.com/OpenDreamKit/planetaryum/blob/master/examples/travis-ghpages.yml`

[17] `https://github.com/OpenDreamKit/planetaryum-example-static`.

[18] `https://github.com/nbgallery/nbgallery`.

[19] `http://notes.quantecon.org/`.

3.2. **Design**

Planetaryum has a modular design, leading to many different types of applications. Its main components are:

- **Readers** are responsible for reading a collection of notebooks from a medium (e.g., folder, git repository, ...),
- **Extractors** are responsible for parsing and transforming the output of a reader to a data stream.
- **Builders** take a data stream and produce an output (e.g., they populate a database or write files to disk); they can be chained to produce many effects at once (e.g., in a full stack application they both populate the database and write the front end files).
- **Front ends** are client side (HTML, JavaScript) applications that take the outputs of a builder and produce a user interface. The library contains a few very basic default front ends, but they are really intended to be developed as separate applications, with their own build chain, invoked by a dedicated builder at build time.
- **Apps** take all the above elements and link them together in a unique app with a well-defined scope. Planetaryum provides a few default apps, but the user is free to write their own.
- The **CLI** is a simple command-line interface that permits invoking the apps and passing parameters to them (through the command line, or through a configuration file).

3.3. **Limitations**

Despite its flexibility, Planetaryum has the obvious limitation of only supporting Jupyter notebooks. Although this reduces its scope, we believe the choice was necessary to make a product that the user would find easy to understand and attractive.

Other limitations, such as not supporting JavaScript-less browsers, are purely technical and could be lifted pending enough demand.

3.4. **Deployments**

For the time being, we have only made test deployments of planetaryum. We now plan to use it for the thematic notebook collections published by OpenDreamKit. In the near future, as the code base consolidates and more use cases emerge, we expect planetaryum to become a tool of choice for larger collections.

4. Conclusion

Planetaryum fulfills and surpasses the original goal of having a tool for maintaining community-curated collections of resources on mathematical software.

We have come to it through a long process of trial and error that has considerably delayed the deliverable. Ultimately, given the multitude of tools trying to achieve similar goals, we felt that it would be a waste of resources to try to build a rich application such as originally planned, with the risk of not seeing it adopted.

Instead we focused on building a modular framework that could be used right away for the simpler needs, such as producing personal or project-related galleries of notebooks, and that could serve as a building block for more complex applications. Towards this end, our next step will be to look for synergies with similar projects, in particular QuantEcon Notes.

Because of this it is hard, for the moment, to measure the impact of planetaryum, but we are optimistic on its future adoption. We plan to advertise it through our usual channels, and we will use OpenDreamKit's notebook production itself as a vitrine for the tool.

Disclaimer: this report, together with its annexes and the reports for the earlier deliverables, is self contained for auditing and reviewing purposes. Hyperlinks to external resources are

meant as a convenience for casual readers wishing to follow our progress; such links have been checked for correctness at the time of submission of the deliverable, but there is no guarantee implied that they will remain valid.