Kevin Sanders

Incorporating Data Visualization into ZENODO

I have spent my nine weeks here at CERN working in the IT-CIS-DLT (Digital Library Technology) department working alongside my supervisor, Lars Holm Nielsen. He is the lead developer on the ZENODO project. ZENODO is a research hosting and sharing website that allows research outputs from all fields of science. Here is the mission statement for the site: “ZENODO is a repository service that enables researchers, scientists, projects and institutions to share and showcase multidisciplinary research results (data and publications) that are not part of existing institutional or subject-based repositories.” My contribution to the project involved incorporating data visualization into ZENODO.

When my supervisor first told me about my project, he gave me a few goals and requirements that I should abide by with my contribution to the project. The first requirement was that whatever I did had to be useful and relevant to ZENODO. Basically what this means is that my addition to ZENODO needed to be something that would be widely used on the site, as well as something that had significant relevance to the site. Another requirement was that my contribution needed to be ‘cool’. Essentially it needed to be something that young researchers such as myself would want to use to upload their research. The next requirement was that my visualization solution had to be web-based, allowing for it to be used on the website. The final requirement was that my project needed to be ready for production by my departure. In this report I plan to talk about the process it took to attempt to complete these requirements.

ZENODO cannot be talked about without bringing up INVENIO, another project that came out of the Digital Library Technology department at CERN. This is because ZENODO is built on top of INVENIO. INVENIO is a free software suite that allows you to run your own digital library or document repository on the web. An example of another website built atop INVENIO is the CERN document server (cds.cern.ch). But how does INVENIO play into my project? Even though ZENODO may have been the main driving force behind adding visualization to the site, if my contribution were to be made in INVENIO, it would also be accessible in ZENODO because it uses the INVENIO software suite. This brings up another point, that by moving my contribution further upstream in the production line, my project has the potential to be used in more sites than just ZENODO, essentially any website built atop the INVENIO software suite.

The early work on my project began by trying to find the best way to fulfill the first requirement: making my project useful for ZENODO. After briefly discussing some of the more exciting options of things to visualize, I came to the realization that my visualization solution would likely be used most by allowing it to visualize something simple like tabular data. There are two main ways to visualize tabular data, either in a table, or as a graph form. Since I find a graph more aesthetically pleasing, I began looking into already existing solutions that allow web-based plotting of CSV files (my file of choice for simplicity's sake). After going through a bunch of Javascript plotting libraries, I discovered ReclineJS, a project developed by the Open Knowledge Foundation. ReclineJS is a Javascript made for handling data, and it has multiple ways to visualize tabular data. It also has a multiview, which allows the user to tab between different ways of visualizing the data, which was

extremely convenient for my project.

The first step for incorporating ReclineJS into my project and later into INVENIO was to come up with a prototype to make sure it served our needs appropriately. The first prototype I put together was outside of INVENIO just to see how the library worked as an external case. This process was made much simpler by the fact that ReclineJS has a plethora of demos that I was able to borrow some code from. By using the code from the multiview demo I was able to efficiently to get a working version of the basic visualization. Because the demo was set up with hard-coded data, the next step for me was find a way to use the multiview with data from a CSV file. Conveniently, this is one of the file types that ReclineJS has backend support for. At this point it was just a matter of figuring out how to use the backend. After some trial and error, I managed to get some CSV code working, borrowing from yet another demo. After combining the two, I finally had a fully working prototype of my visualization solution outside of INVENIO.

The clear next step was to inject what I had so far into a working version of INVENIO. This is where my education of the INVENIO infrastructure began. I needed to find where different things go in the installation that also allowed my code to work. After some tinkering I managed to find a good enough place to drop my code into (as well as the ReclineJS library code). All seemed well and good until I discovered a bug where the way that INVENIO dynamically loads some stylesheets interferes with how ReclineJS is put together, causing my prototype to only function in some hardly used browsers like Opera. After digging deep into the depths of the ReclineJS developer forums I found that it was an issue with their code and also found a few possible workarounds. After trying to find an

elegant way around the problem to no avail, I had to use the solution that involved wrapping all the multiview demo code and forcing it to wait for the page to load before loading the visualization. It wasn't a very nice solution but it was the only one that worked. Once I fixed the problem my prototype was finished.

With a completed prototype that demonstrated proof of concept and that ReclineJS would work in INVENIO, I could finally start implementing the final product. The implementation process was really where I began to learn a lot about how to work with INVENIO and how to use the many tools it encompasses. INVENIO has a good number of modern web development tools that are very useful and make the whole process a lot more efficient. Some of the ones I got most familiar with included Flask, Jinja, Bootstrap, and general Python. Flask is a micro web development framework for Python. The main use I had for it was rendering HTML templates and passing variables to these templates to later make the visualization possible. Jinja is a modern and designer friendly templating language for Python. Essentially it allowed me to use Python code in my HTML templates for things like working with variables and conditionals to determine which HTML code to render. The Twitter Bootstrap is a front-end framework that allows fast and easy web development. The Bootstrap came in handy when trying to add an aesthetic feel around ReclineJS with buttons and headers. In the end it made it look much nicer than I would have managed writing my own code. And Python is used all over the place in INVENIO; it is the main language used for all the backend processing.

The actual implementation of the data visualization was done in the form of plugins. The decision to use plugins was made to allow for easy overriding and the later addition of

more ‘previewers’ for different file types. Because ZENODO already uses Google Docs Viewer for PDF files, it made sense to use plugins because it made it trivial to add the viewer to the visualization suite. Making these plugins for different previewers was achieved using Flask and an INVENIO tool called pluginutils. The technique that I used with the plugins was to find the file type, then loop through all the plugins (using pluginutils) to see which plugin, if any, can preview that file type. Once a match is found, it’s simply a matter of using Flask to render the template associated with that plugin and passing the file information to that template where the data is then visualized. This process works well because no code needs to be changed when adding another viewer, the plugin just needs to be put in the INVENIO install and the existing code takes care of the rest.

Later in the implementation, the idea came up to make these visualizations embeddable. It would be pretty nifty if someone could just click a button and then be able to put the visualization into their own website. The simplest solution for me was to add an embed button, that when clicked, has a popup with code needed to embed the visualization into a website. Then it’s just a matter of copying the code into another website to use the visualization. Doing this was another use for the Twitter Bootstrap. As well as having code for buttons and headers, Bootstrap contains code for things like modal pop-ups, which I used with the embed button. I learned a great amount using the web development tools contained in INVENIO and was successful in laying down the initial infrastructure for data visualization. Soon enough, ZENODO will have my code integrated, and then hopefully at some later point, it will be possible to preview any type of data that someone would want to upload to the site.