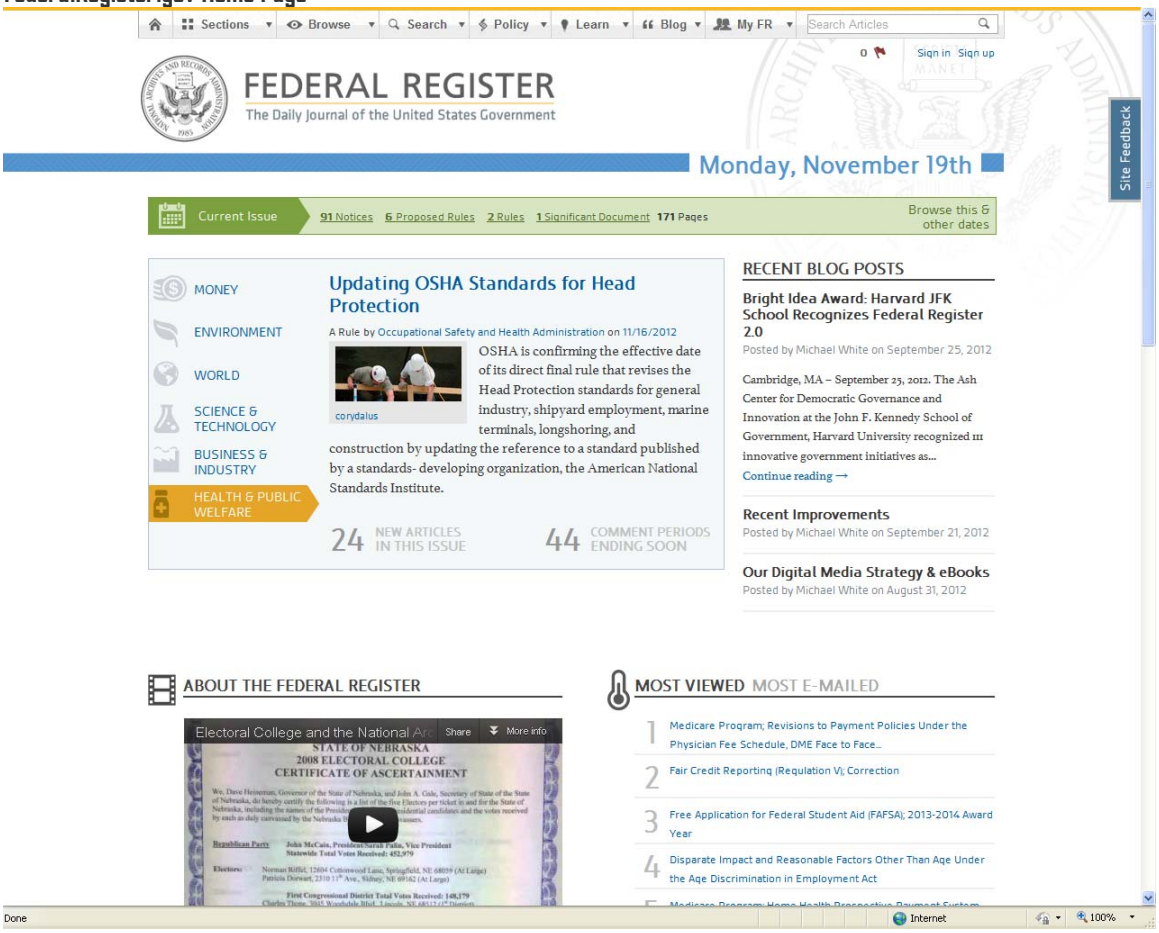
FederalRegister.gov API Case Study

Background

On July 26, 2010, the 75th anniversary of the Federal Register Act, the FR2.0 project was launched on FederalRegister.gov. The goal of project was to make the rich content of the Federal Register available in an easily consumable way. FederalRegister.gov builds on the bulk XML data provided by the Government Printing Office, and layers in content from various other federal sources (including OMB resources on RegInfo.gov and Regulations.gov) in order to provide a deeper context to the regulatory content.

Open accessibility has always been a high priority for the project. This is because the Federal Register Act was passed to provide transparency to regulatory documents, and FederalRegister.gov has its origins in an open source public project. Additionally throughout the design and creation of FederalRegister.gov, we made decisions with an eye to releasing an API so other developers would not need to replicate the time-consuming work of integrating various federal sources into a coherent regulatory picture.

FederalRegister.gov Home Page



API

One of the difficulties for anyone seeking to utilize government data is to match up datasets from various federal sources. Identifiers such as field names and agency name references often are not identical. Further, the data is often complex and requires a fair amount of understanding of agency processes to create a coherent whole.

On August 1, 2011 the FederalRegister.gov API was released to address this problem and make it easy for anyone to use regulatory data as machine readable material. The API provides access to all data found on FederalRegister.gov, including all Federal Register documents from 1994 to the present. Because ease of use was our primary goal, we implemented the following tenets:

No API Keys

In our view, API keys can create an unnecessary barrier to rapid experimentation with our public data. We are able to track our API usage via logging mechanisms on our servers and already have infrastructure in place to mitigate any sort of excessive requests. The benefits of using a simple REST-ful API format are that any user can easily try it in their browser (no SOAP that requires complicated XML to be POSTed around, no special headers, etc). The response to our no keys policy from the development community has been extremely positive (http://news.ycombinator.com/item?id=2839137).

Allow the user to determine the content they want

By connecting the API to our robust full text search engine, we have been able to go beyond merely providing access to tables of data. We allow users to access the data by slicing it in any way they desire. By default, a generic set of fields is returned, but the user can easily request only particular fields. This is particularly valuable on mobile data connections where users are charged for data usage and data speeds are much slower. Further, a user can select documents using any of the many filters available on our advanced search page (https://www.federalregister.gov/articles/search#advanced)

Provide robust documentation

We released interactive documentation to help users explore our data, and to give more novice users an easy way to build API queries. For simple applications, users can take advantage of the dynamic web interface available at https://www.federalregister.gov/developers/api/v1 to visually build a query that they can then use in their mobile app or on their website.

Programming language specific libraries

To make it easier to integrate our API into applications, we released a Ruby library as we launched our API. At launch we were quickly made aware of other developers creating libraries in other languages. Again by removing the complexity that API keys or SOAP interfaces impose, those libraries are much easier to build – making it more likely that the wider community will participate.

Make use of the API internally

Not only does this ensure that your API is of high quality and does not become abandoned, it also allows for reduced long term development and maintenance costs. The MyFR section of FederalRegister.gov was created as a separate web application that communicates with the main application only via the API. Additionally we leveraged the API on various pages of the site (search results, Executive orders, etc.) to provide data directly to users in CSV format. By using the API internally we are able to modularize our code base, making it much easier for long term development while also reducing the sort complexity that causes inerrant and difficult to find bugs.

Response

The response from both developers and the open government community was overwhelmingly positive (for an interesting discussion see, http://news.ycombinator.com/item?id=2839137).

We've seen the creation of various applications from simple one page web sites to full fledged applications that make robust use of the API on a daily basis. On the simple end of the scale, Etienne Benson of the Max Planck Institute in Berlin created an application to track Polar Bear regulations. He writes, "I've taken advantage of the blissfully simple FR2.0 API to create a tool that automatically grabs all FR items that mention polar bears, displays them in a nicely formatted list, lets you browse forward and backward in time, and links back to the full text on the FR2.0 site."

On the more complex side, the Sunlight Foundation utilizes the FederalRegister.gov API as part of their recently released Scout application. Regulatory content is pulled from the Federal Register API on a daily basis and combined with other federal and state data to help create a larger picture of government actions. As described by Sunlight, "Scout allows anyone to subscribe to customized email or text alerts on what Congress is doing around an issue or a specific bill, as well as bills in the state legislature and federal regulations".

We are also aware of a financial services firm using our API to provide small banks with notifications of regulatory action that concern them directly. This is a great example of how the robust search functionality integrated into the API enables more complex slicing of our data for a particular purpose.

Recommendations

1. Make the barrier to finding and experimenting with your API as small as possible.
2. Allow developers to specify the fields returned - this can be very helpful when building mobile apps, especially when the data available is quite large.
3. Robust documentation is important and can be simplified by providing easily understood field names. Consider using something free and open source like IODocs (https://github.com/mashery/iodocs#readme) to quickly stand up interactive documentation.
4. View your API as an opportunity to make legacy data easier to consume. For example, your API representation can help make using your data easier by presenting it in a more understandable way that hides internal acronyms, etc.
5. Utilize your API on internal projects - this ensures that you identify and fix bugs quickly and saves money by reducing app complexity. Additionally it makes it easier and more natural to identify features that are beneficial in practice rather than ones that simply sound appealing.

References

FR Developers Page: http://federalregister.gov/learn/developers
IODocs Project: https://github.com/mashery/iodocs#readme
FR Interactive Documentation: https://www.federalregister.gov/developers/api/v1