The PhEDEx Next-gen website

The architecture of the next-gen website

The next-gen prototype: original

The PhEDEx® data service® has expanded considerably in power in the last few years, but the original website® has not profited from this because of its different codebase. We wanted to make available to users in a visual way, not just programmatically, the full richness of monitoring capabilities that the data service provides.

We also wanted to separate the codebase for the website, which would reduce redundancy and avoid possible confusion should the website and the data service provide information differently, or in a way the user is not expecting.

Architecture of the next-gen website

The next-gen architecture follows a well-defined model. There are three main components, the sandbox, the core, and one or more modules. Modules can be further enhanced by decorators, which are themselves just a specialised form of module.

The sandbox manages communication between components. They notify the sandbox of events, and listen for notifications from other components or from themselves, registering a callback function for the sandbox to invoke. Using the sandbox to send themselves notifications instead of calling their own methods can mean they can yield the CPU, making for a more responsive application.

The sandbox generates javascript events and handlers for them automatically, and maintains a list of which components have subscribed to which events. Components need only agree on an event name and a signature (arbitrary set of objects) that parameterise that event, the sandbox takes care of everything else.

The core is responsible for creating modules, for driving them through their lifecycle, and eventually for destroying them. It calls module methods directly, or notifies them via the sandbox.

The core contains the application logic, but does not know what data the modules use, or what services they provide to the user.

Modules provide specific functionality or representations of data in the browser. Several modules can co-exist and can interact through the sandbox. Modules never call each other, or the core, directly.

Decorators enhance modules by adding new functionality in a generic way. This allows customising modules by adding extra features without modifying the module code. Decorators are allowed to call the methods of the module they belong to, but can also interact via the sandbox.

The filter-panel is a decorator which is driven entirely by metadata in the module. The metadata describes the data structure that module represents.

The User Experience workshop

Initial reaction to the prototype was not good. We had intended to offer something new, but what the users really wanted was what they already had, with a number of improvements.

Many users found the site non-intuitive, or overwhelming in the amount of information it provided.

So we held a User Experience (UX) workshop, to explore ways of enhancing the existing design with the next-gen framework.

The UX workshop ran for 5 days with 9 participants. Two days were spent exploring the results of a detailed survey which was filled out by volunteers from the collaboration. 58 responses were received.

Two more days were spent wire-framing, with paper and whiteboard. This was focused on specific pages of the existing website, one by one.

The new next-gen website

We started with the most common pages and enhanced them first.

This gave us feedback from a wide audience in a more controlled manner, and the users have welcomed the improvements.

This allowed us to eliminate a lot of the redundant code from the old website in a realistic timescale with available manpower. It also avoids a ‘bet the farm’ approach to releasing the site.

The next-gen architecture adapts well to implementing new pages or to enhancing an existing website. The existing site needs minimal adaptation to allow the next-gen framework to be used. The process of replacing the old website is ongoing, and on track.

The original next-gen prototype is still used by some people, such as the central site personnel, who make regular use of its features.

References

1. CMS Data Transfer operations after the first years of LHC collisions, CHEP 2012
2. Data transfer infrastructure for CMS data taking, ACAT 2008
3. The PhEDEx data service, CHEP 2010
4. https://cmsweb.cern.ch/phedex

The PhEDEx Next-gen website

awildish@princeton.edu