The HEP Software and Computing Knowledge Base

T Wenaus
Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA
wenaus@gmail.com

Abstract. HEP software today is a rich and diverse domain in itself and exists within the mushrooming world of open source software. As HEP software developers and users we can be more productive and effective if our work and our choices are informed by a good knowledge of what others in our community have created or found useful. The HEP Software and Computing Knowledge Base, hepsoftware.org, was created to facilitate this by serving as a collection point and information exchange on software projects and products, services, training, computing facilities, and relating them to the projects, experiments, organizations and science domains that offer them or use them. It was created as a contribution to the HEP Software Foundation, for which a HEP S&C knowledge base was a much requested early deliverable. This contribution will motivate and describe the system, what it offers, its content and contributions both existing and needed, and its implementation (node.js based web service and javascript client app) which has emphasized ease of use for both users and contributors.

1. Introduction

The HEP software and computing world today is a rich and complex amalgam of HEP in-house software and facilities together with resources drawn from the wider, largely open source, community. Having sufficient awareness of this landscape to take full advantage of it in our work is difficult, and yet of great benefit in leveraging what's already out there. As HEP software developers and users we can be more productive and effective if our work and our choices are informed by a good knowledge of what others in our community have created or found useful.

The HEP Software and Computing Knowledge Base, hepsoftware.org [1], was created to facilitate this by serving as a collection point and information exchange on software projects and products, services, resources, training, publications and computing facilities, and relating them to the projects, experiments, organizations, institutions and science domains that offer them or use them.

It was developed as a contribution to the HEP Software Foundation (HSF) [2], for which a HEP S&C Knowledge Base was a much requested early deliverable: an initial charge to the HSF was to "provide a system for facilitating information exchange". While the first prototype of hepsoftware.org predated the establishment of the HSF, the project was given impetus by the HSF's priorities. A proposition was accepted at the HSF's formative workshop at SLAC in early 2015 that HEP S&C should have a grass roots information gathering and exchange hub to

- Facilitate collaboration by increasing awareness of resources, projects, activities
- Provide a place to look for solutions, learn from the choices made by others
- Promote awareness of software and tools from outside our community, e.g. open source
Key to the objectives of the knowledge base (KB) is its grass roots nature. The knowledge base must be a resource built and maintained by the community itself; only in this way can it become comprehensive and current enough to be fully useful. There is a Catch 22 to overcome in this: it’s only useful if it’s populated, and people will only help populate it if it’s useful. These considerations were major drivers in the design, to offer a tool able to entice people to contribute by making it quick and fun (or at least pleasant) to do so.

2. Design precepts
The principal design precepts for the knowledge base were:

Employ an entity-relation model. With the principal purpose of the knowledge base to describe entities (software, software categories, organizations, etc.) and the relations between them (dependencies, categorizations, ownership, etc.), the E-R model is perfectly suited, as has proven to be the case in the implementation. The knowledge base is all about making connections, making relations at least as important as entities themselves. Relations embody the knowledge built up in the KB: much of the most useful experiential knowledge is in the interconnections. You don't need a knowledge base to tell you what Jenkins is, or the Apache 2.0 license, but a quick and convenient way to learn who in our community is using them is valuable and not amenable to googling.

Make it fast and highly navigable. With discovery and exploration another principal purpose of the knowledge base, the design puts a premium on presenting rich information that can be navigated and explored quickly. The design follows something of the philosophy of the PanDA [3] monitor [4] (for which the present author is the principal developer) in presenting dense and richly interlinked information, proven popular and effective among users seeking out information and interconnections on the PanDA workload manager's operation, but with the essential difference that the intelligence and the data reside on the client side in the knowledge base, making possible much greater speed and interactivity.

Users are also creators. With a principal requirement of the knowledge base being that the community itself populates it, it is essential that the design support easy transitioning between browsing and editing, and easy addition and editing of entries. Furthermore with all registered users having edit rights to the full knowledge base, thorough journaling with rollback is necessary, as well as frequent backups. The registration itself had to meet important requirements. Registrants should be responsible actors in the system, so they should be identified, not anonymous. Registration should be easy and leverage the identity and trust of large organizations. Accordingly registration is based on the user's email as obtained from OAuth2 based registration with Google, GitHub, Dropbox, Amazon, etc.

3. Implementation
The implementation went through three generations of prototyping to finally arrive at the desired functionality and performance in the third generation. The development history in brief:

V1, 2014: The first version drew on the ATLAS PanDA monitor, sharing some of its code base. It used a Django based Apache web service backed by MySQL, an xml based entity-relation data format that could be used to populate the knowledge base in bulk, and a web form based editing interface in the browser. As a first prototype it validated the general architecture of the E-R model but the complexity of Django and its design ill-suited to developing a client-rich app excluded it as an implementation approach.

V2, 2015: The second version integrated two objectives, the knowledge base and the HSF website. Drupal was chosen as the basis for implementing a website that combined the two functions. However
the number of Drupal extensions necessary to approximate the desired knowledge base functionality was very large and made the service slow and not scalable.

**V3, 2016:** The third and present version utilizes the rapidly growing technology of a rich client web application employing javascript both client and server side, enabling intelligent and efficient operation on both sides. It also enables rich, responsive client behaviors largely autonomous from the server when data volumes permit (as they do in this application). With this approach the design precepts were realized fully, with essentially unlimited scaling headroom and with broad potential to leverage the rich web app toolset to expand the functionality.

**2017+:** While the good match of capability and required functionality in the present version validates the implementation approach, the implementation itself (by a javascript neophyte) presents opportunity for improvement. It does not employ client-side one of the many frameworks available; after trying out several, using none was found to give the greatest control and flexibility over the end result, but a judicious choice of framework would probably improve the economy and maintainability of the code. The app could benefit generally from a re-engineering to make it more maintainable and production quality; the HSF is trying to identify effort for this.

The KB software is factored into a knowledge base core and a number of applications implemented as lightweight modular extensions of the core, of which the HEP S&C knowledge base is one. Other applications include R&D prototypes in applying the knowledge base to problems in ATLAS metadata management and distributed computing, and will not be further discussed here. The core code base is about 10k lines of javascript (server and client); the HEP S&C KB layer is about 2k lines. The core implements the E-R model via Entity and EntityRelation data structures mapped to a relational DB (MySQL) in the server back end.

The core supports moving data between the client and server in two ways: standard ajax (using jQuery [5] for this and much other basic functionality), and CouchDB/PouchDB [6], [7]. The latter supports a publish/subscribe model appropriate when the system manages more data than the client can absorb at one time. However in the S&C KB app, the data itself is very compact (purely text, a few MB) and the client in the browser can readily accommodate the data, to the benefit of the client’s speed and interactivity, so only ajax is used at present. CouchDB/PouchDB can be used in the future if needed. Amazon S3 is also supported but is not presently in use. For knowledge base R&D purposes, Oracle, Cassandra and Amazon DynamoDB are also supported.

The server is implemented using Node.js [8] and uses Express [9] as a framework. The server is entirely core functionality; S&C KB app specifics are limited to configuration. In addition to MySQL [10] as the back end repository and CouchDB as a client/server communication channel, it uses the Redis [11] in-memory database as a transient cache and session state repository. The Passport [12] package is used to implement OAuth2 authentication via Google, GitHub, Dropbox, or Amazon in a user-friendly way, with the user identified by their email address which tags their edits in the system. The server also provides the interfacing to external services, in particular Google, supporting the automatic ingestion of the HSF’s Google based activity calendar [13].

The client is implemented in generic Javascript with wide use of jQuery for basic functionality. As a rich client web application, the client is where the functionality and expressiveness of the app is built, as described below in the feature summary.

The hepsoftware.org software and content is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. The knowledge base software is not yet publically available, pending refactoring to purge ‘secrets’ from the code base.
4. Feature summary

The KB presently catalogs software packages, categories, experiments, science fields, organizations, institutes, events and resources. Weblinks may also be added, and can be dressed with comments, tags, relations to KB entries etc. Various views are available, e.g. lists of software categories, categories with their associated software, and a list of all software. Similarly experiments can be listed together with the software they use, or conversely software can be listed indicating the experiments that use it. The same applies to sciences, organizations, institutes, and resources. Sorting can be temporal or alphabetic.

![Figure 1. The HEP S&C Knowledge Base](image)

The app’s layout is organized as listings on the left, detail in the middle, edit and search on the right, as shown in Figure 1. As lists are navigated by the user, the center detail panel updates instantly, making scanning very fast. The top bar determines what to list: Software (and Software Categories), Experiments, Sciences, Organizations, Institutes, Resources (which are Facilities, Services, Publications, Training, Careers and Funders), Links (web links), Events, People and Tags. When an entry is selected, related entries are shown and highlighted. Supported relations include associated with, used by, uses, category, science field, and more information. Entries can carry a large variety of attributes including website, contact, various sorts of documentation and information (wiki, reference, document, presentation, link), repository information (repo link, GitHub, Bitbucket, issue tracker), license, communication information (forum, blog), events and event series, training, various social media, testimonials (for qualitative star ranking), and job listings. Furthermore, entries can be associated with Tags, either existing or user defined, providing unlimited scope for creating associations and classifications.

While viewing an entry, a click or a keystroke can activate editing it. All registered users are able to edit and add any entry. A full revision history is maintained, and older versions can be viewed and reverted to. Content is written in Markdown [14], and in edit mode the middle detail panel becomes a live preview panel showing the edits in progress. The greatest weakness of the earlier prototypes was in the slow, laborious editing; in the present version it is fast, responsive and intelligent. Auto completion is used throughout to expedite editing. Auto save (to the server) is used throughout, and edits are cached locally until a server update succeeds. Multiple entries can be edited at once.
Scanning listings with the mouse gives an instantaneous view of the associated entries, until an entry is selected with a click (highlighting its name in red). The center panel then remains fixed on the selected entry. It can be de-selected by clicking the highlighted name in the center header or left list, or with ctrl-x. When an entry is selected, related entries are highlighted in light blue in the left window lists.

The body of the entry is written in Markdown. Where it is long and structured enough to have sections, a Show Sections button in the header can be used to show a table of contents in the left window. Besides the body text describing the entry, it includes the references and relationships such as links to website, repository, documentation, associated science fields, 'uses' and 'used by' relations, and so on. The full scope of the presently supported relationships can be seen in the entry editor.

Notifications will occasionally appear (briefly) to inform what the app is doing (or refusing to do). Keyboard shortcuts are provided for frequently used functions and are documented in the Help panel. The search panel provides instantaneous comprehensive search with autocomplete on either entry title or everything else. With no search selection, it lists all entries, either by most recently modified or alphabetically. Scanning the list instantaneously displays the entries.

5. Status
The knowledge base resides on an Amazon EC2 server sponsored by Brookhaven National Laboratory; the HEP Software Foundation website resides on the same server. Backups are performed every few hours and kept daily. The system presently contains about 700 entries and 1000 associations, with contributions from about 25 people. While non-negligible this is well below the desired level of both content and contributions. Comments on ease of use are generally favourable but translating that into more extensive contributions proves difficult. Development of the system has been at a low level since Spring 2016 because of other commitments; the HSF is trying to identify new effort.

6. Next steps
The list of possible improvements is long; here follow a few higher priority ones.

- Extend OAuth2 support to include CERN SSO. Information from CERN is that they now offer this.
- The most frequent functional request: support People as full fledged entities in the system, such that people can relate themselves to software, experiments, organizations etc., and can create a profile.
- Provide a notification mechanism for new content (RSS and/or email).
- Complement the present support for ingesting Events from the HSF Google calendar with the reverse functionality: propagating Events registered in the knowledge base (where they are associated with Entities like organizations, experiments etc) to the HSF calendar.

7. Conclusion
The HEP Software & Computing Knowledge Base at hepsoftware.org is a collection point for HEP related software projects and information on HEP software and computing. It can be used to look for existing software, see what others are using, make a new project known to the community, and inform the community of the software you and your experiment use. The knowledge base is a project of the HEP Software Foundation. Everyone in the HEP computing community is encouraged to contribute to the content, by adding software, adding experiments, adding interrelationships describing who uses what, and so on. The more complete and reflective of community knowledge, experience and opinion the content is, the more useful it will be. Feedback is encouraged. Send it to hepsoftware@gmail.com or to the author, Torre Wenaus at wenaus@gmail.com.
8. Acknowledgement
The author thanks the many people who have commented on the knowledge base and suggested improvements. This work was funded by the U.S. Department of Energy, Office of Science, High Energy Physics under Contracts No. DE-SC0012704, No. DE-AC02- 98CH10886 and Contract No. DE-AC02- 06CH11357.

References
[1] HEP S&C Knowledge Base, https://hepsoftware.org
[2] HEP Software Foundation, http://hepsoftwarefoundation.org
[3] PanDA workload management system, http://news.pandawms.org/panda.html
[4] PanDA monitor, http://bigpanda.cern.ch
[5] jQuery, https://jquery.com/
[6] CouchDB, http://couchdb.apache.org/
[7] PouchDB, https://pouchdb.com/
[8] Node.js, https://nodejs.org/en/
[9] Express, http://expressjs.com/
[10] MySQL, https://www.mysql.com/
[11] Redis, https://redis.io/
[12] Passport, https://www.npmjs.com/package/passport
[13] HSF community calendar, https://goo.gl/g9Kys9
[14] Markdown, https://daringfireball.net/projects/markdown/