OPEN SOURCE APPLICATIONS FOUNDATION

Data Framework Project

This is the top-level page for the Data Framework Project.

Project Summary

We want Chandler to be good at handling both structured and un-structured data. Chandler's data handling framework should be less rigid than say a relational database, but allow the user to add more structure than is possible with a spreadsheet or a word processor. The goal of the Data Framework Project is to design and build some data handling infrastructure for Chandler. The data framework will need to handle both structured and un-structured data, and it will have to meet the needs of both parcel programmers and end-users.

Active Contributors

- Owner: Brian Douglas Skinner <skinner@osafoundation.org>
- Active Contributors:
    - Mitch Kapor -- vision, requirments
    - Katie Capps Parlante -- python mapping prototypes
    - John Anderson -- architect
    - Andi Vajda -- repository

Project Status

- June 2003
    - Andi working on bottom-up implementation of the data model building blocks
    - Brian working on top-down formal representation for a proposed data model
    - Katie working on thorny issues in the schema (e.g. users, contacts, personas)

Release Planning

- Goals for 0.2 release (mostly cribbed from DotTwoPlanning)
    - "Provide a first substantial iteration of the Chandler data model and create the table outline widget that allows for

complex inter-relationships of information spanning multiple domains (the "soul of agenda" feature)"
    - "end-to-end data -- ...demonstrate end-to-end data handling. Have support for only some fairly basic level of data expressiveness, but for each aspect of that expressiveness, have it be handled consistently across all the parts of Chandler"
    - "Python mapping -- Have a mapping to Python objects that can represent Chandler data"
    - "Add and edit attributes on-the-fly for any information type."

- Goals for 0.3 release
    - Offer some guarantees about data preservation -- be able to import 0.3 data into 0.4
    - @@@ ...

- Goals for 1.0 release
    - @@@ ...

Bugzilla

- @@@ -- in bugzilla, we have comingled the "data framework" work with the "Chandler calendar PIM schema" work, so you can't search for just one without the other
- All open bugs under the heading "Data: PIM Schema"

Vision Documents

- Mitch's weblog -- 24 Oct 2002
- Mitch's weblog -- 15 Dec 2002
- General Information Management
- ..."data types are richly integrated"
- ...ad-hoc attributes
- Data Framework Vision

Design Documents

- Data Model document nexus
- Observable Queries proposal
- Use cases -- none
- Prototypes -- none

Decisions

- still working on drafting an initial data model proposal
- tentative proposals recorded at Data Model Feature Summary

Open Issues

- what sort of interoperability with RDF, RDFS, OWL?
    - see also: Katie's notes from the RDF Calendar Chat, 11 June 2003

Sub Projects

- Data Model

Related Projects and Dependencies

- Repository Project
- Document Architecture Project
- Python Binding Project
- Unified API Project
- RAP Project
- Chandler PIM Schema Project
- Notification Manager

Risks

- May create a data model that is too complicated
- May create a data model that is too inflexible
- May fail to achieve interoperability with RDF and other open standards
- May fail to achieve interoperability with relational databases & other de-facto standards

Review Status

- Usability review -- not ready for review
- Security review -- not ready for review
- Performance -- not ready for review
- Schema reviews -- not ready for review

More Info

- Data Framework Project Bookmarks

History

- May 2003 -- initial data model design meetings with John, Andi, Katie, and Brian.
- @@@ -- old design docs use old terminology

Contributors

- BrianDouglasSkinner - 24 Jun 2003

Discussion

(none yet)

Mitch Kapor's Weblog[1]

October 30, 2002

Working in a New Way

For the past few days an ardent discussion on the OSAF design mailing list has revealed there is no consensus about the best way to collaboratively gather and sift design input about Chandler, a task which has become quite important.

I've learned since yesterday about the wiki, the twiki, the zwiki, and the wikipedia, all variants of an open source tool for collaborative editing. Each has its partisans and critics and each is used, in varying ways, by open source projects. We are investigating.

In a conventional corporation, there are a basic set of ground rules about how activity is organized. Corporations are still ultimately hierarchical. Some pyramids of power are steeper, others flatter, but they all point to the top. Anyone who works in the business world absorbs a set of default rules about how workplaces operate, e.g., your boss tells you what to do, but you don't tell your boss what to do.

As an open source project, we are experiment in progress. Like every other project we have to determine how decisions are made and who has what kind of power. Other projects have pioneered various methods of organizing their own activity, i.e., who can submit code for inclusion and who can commit it. There's a lot to learn from them and there is not yet any general consensus.

I had a fascinating conversation yesterday with Mitchell Baker, the general manager of mozilla.org, after which I began to appreciate the scope of process issues we are going to have to deal with. Active and successful community participation will require careful choice and use of the communications vehicles. I've already seen that the openness of mailing lists coupled with relatively wide membership and the willingness of participants to express themselves means that more than one side of most every issue is going to be brought up, no matter how small the issue.

Case in point: I received an email from the design list which I wanted to reply to. I use the reply command of Eudora. The new message is addressed by default to

[1] From http://blogs.osafoundation.org/mitch/2002_10.html#000026

the person who posted the message I'm replying to, not to the list as a whole, which is what I want. I make a note to myself let Morgen, a developer who's currently doubling as system administrator to see whether the Mailman program we use can be reconfigured to fix what I regarded as a problem. Before I got around to sending it, I read perhaps a dozen postings to the list discussing what the correct behavior should be. Surprisingly, there were articulate and strong arguments on both sides. Not only that, but there are already lengthy essays on this posted in various places on the net which were cited.

Looking into it, I discovered there are substantive points about how replies to lists should form the To: address. One the one hand, since the previous message was sent by a poster to the list, a reply to the message ought to be sent by the default to that poster, as a reply is meant to go to the author, not the recipients, of the previous message. If I want to include the recipients, I should use the Reply to All command. On the other hand, what most people probably want to do is to reply to the list (and perhaps copy the author), in which case the "proper" solution doesn't do what is wanted and requires extra work. So why not munge the headers of the posted message such that when I do want to reply, the To: address of the new message is the list. For the rest of the gory details, you could start here

In a world in which all points can be discussed endlessly, it becomes important to have generally agreed-to processes by which decisions can actually be made and respected. Whether munging reply-to headers is really harmful or not isn't the point. It is that it's a Brave New World out there and we better get used to it. What's new here for me is not the endless discussions of mailing lists, but their impact on a project which is trying to work in an open and participatory fashion. And everything I discussed here is just the tip of the iceberg.

Posted by mitch@osafoundation.org at 04:31 PM

Mitch Kapor's Weblog[2]

December 29, 2002

Making Design Decisions

[The text below is a draft of material to be included in the Chandler Community Wiki, which we hope to deploy next week. We're beginning to focus on the pragmatic aspects of the project. This is a taste.]

Making Design Decisions: Some Principles

This area [of the Wiki] is for discussing the principles which will guide decision-making about Chandler's design. This is not meant to be all-inclusive list, just a start. All of these principles are subject to comment, criticsm, and improvement. You may wish to propose and give reasons for your own.

The first set of principles have to do with making decisions about which features and capabilities need to get into earlier releases and why. It thus forms a partial rationale for scheduling the work to be done.

It's often easier to state a principle than to apply it, but you have to start somewhere.

1. Implementation must be sequenced

It's not a real project until commitments are made to defer some capabilities. Doing everything at once is not an option.

2. First, provide core functionality users expect

If a feature is in common use in today's PIM's, then it needs to have its equivalent in Chandler. In order for the product to be adopted, it must have the critical mass of features people expect.

3. Lead with Chandler as an application, not a platform

[2] From http://blogs.osafoundation.org/mitch/2002_12.html#000092

The Chandler project is intended first and foremost to deliver an application suitable for daily use to handle email, calendar, contacts, and tasks. We believe Chandler will embody components of great potential usefulness to many applications such as the repository and the view manager, so it is fair to think of Chandler as a platform also.

We are making a clear choice to lead with the application-oriented aspect of Chandler, rather than the platform-oriented aspect. We believe that application adoption will provide great leverage for the entire project

Moreover, we want to create an application we ourselves would like to use. No such application exists today and the felt need is urgent.

Understanding application requirements will help shape the platform to be more suitable. (I am indebted to Andy Hertzfeld for this observation.) An implication is that it would be useful for platform development to be working on a second application simultaneously.

We don't know if we will have the resources to work on a web client for Chandler at the same time as the PC version, but if we do it will help make Chandler a better platform.

We will do the best job we can to think through platform issues form the outset and embody them in our architecture, even if we have to defer implementation of some things necessary for Chandler as platform.

4. Begin with Chandler's PIM functions: email, calendar, contacts, tasks; add full knowledge management later

Accept for the moment, if you will, that email, calendar, contacts, and task management comprise the PIM aspect of Chandler. Then, think of Lotus Agenda and ECCO as examples of tools for ad hoc knowledge management with some PIM functionality.

We intend to develop the PIM aspect of Chandler first with sufficient fullness to enable it to be in daily use. Chandler as PIM will be built with the rich semantics and flexible views found in the ad hoc KM tools, and there will certainly be a base set of knowledge management features, but the full development of Chandler as ad hoc KM will come after Chandler as PIM.

PIM's are universal productivity tools, while KM's are not (though many of us wish they were). As in the principle of "Chandler as application first" the adoption of Chandler as PIM first will provide leverage that will benefit the entire project.

No doubt having to wait longer for the complete delivery of Agenda-like and ECCO-like functionality is going to be disappointing for many of the loyal and faithful.

5. A few killer features are needed from the outset to make adoption worthwhile

Chandler must do a few things radically better. Candidates include: sharing; robust, transparent security; agent architecture; Agenda-like flexibility; [add your favorite here]

Killer feature ideas must be sufficiently well-developed as ideas to be willing to place bets on them. This is a key priority.

Posted by mitch@osafoundation.org at 10:23 AM