Chandler Release 0.1

Purpose and Goals

Purpose

The purpose of releasing version 0.1 Chandler source code is to provide an architectural and technical overview of Chandler, give the community a chance to review a skeletal framework and tentative APIs, and to provide more details about future Chandler plans including a few cool features to give a glimpse of what is possible. In general, this is a chance to show that OSAF is "for real".

Goals

OSAF hopes that the Release 0.1 will be complete enough and documented well enough for interested, proficient developers to follow instructions and build the release without asking us for further assistance. Users should be able try out a very initial skeleton of Chandler without further assistance from us, and developers can gain a better understanding of the Chandler architecture, framework and APIs select components of Chandler.
"Throw away" code will be kept to a minimum and well-documented as such. Developers will be able to use this release to contribute to intelligent architectural discussions, possibly highlighting architectural flaws. And brave, risk-loving developers may try writing sample parcels and point out problems that need to be addressed. The 0.1 release will include TAR/Zip files of binary and source code for all components, including Python and wxWindows, on Linux, Mac and Windows.

What's not planned -- topics we'll address, but not in the 0.1 release

Release 0.1 is not intended to demonstrate a complete feature set, a final UI, security mechanisms, a final database or schema, or be ready for end-user deployment

<**top**>

Components of the 0.1 Release

The 0.1 release is a very early, partial implementation of parts of Chandler. It's intended to give the flavor of Chandler and get valuable feedback on architecture and APIs. It is not intended to capture more than a tiny fraction of the product we have planned. The descriptions below will make more sense if the reader is familiar with the concept of a Chandler Parcel. The 0.1 release includes some basic infrastructure pieces as well as the beginnings of several parcels.

1. Parcel Framework

The foundation of the application, the parcel framework provides common services to the parcels. For this release we've focused on one particular type of parcel: Viewer Parcels. A Viewer Parcel is an area of the desktop user-interface which can view various kinds of items, for example, the calendar view for viewing calendar event items, an email view for viewing email items, a contacts view for viewing contact items, or a generic table view that can view items of any type. The parcel framework also manages the Parcel Independent UI.

2. Parcel Independent UI

The Parcel Framework UI provides user interface elements common to the application, independent of a particular Viewer Parcel. Elements include:

- Navigation bar, the mechanism used to navigate through the application's views
- Menu items, both menus that are common to all parcels and parcel specific menus
- Sidebar, shortcuts to let the user switch between parcel views

3. Common services for Viewer Parcels for the 0.1 Release include:

- A mechanism for loading Viewer Parcels
- A mechanism for Viewer Parcels to use menus
- A sample parcel to exercise the Parcel Framework and be an example for Viewer Parcel writers. The sample parcel will display all items in the repository, in a generic table view.
- View persistence mechanism, used by both Viewer Parcels and the Parcel Framework. The exact mechanism is likely to change for future releases.

4. Calendar Parcel

The 0.1 release has rudimentary functionality for this parcel, including:

- Columnar View
- Create, move, edit, resize, delete simple events in the Columnar View
- Month View
- Ability to switch between Columnar View and Month View
- Basic navigation: jump to previous week/month, next week/month
- Rudimentary Month Navigator (uses a wxCalendarCtrl for simple navigation)
- View persistence (the application will remember the state of the calendar views)
- Event persistence (the application will remember the events that were created, in a rudimentary repository)

5. Contact Parcel

This parcel also has rudimentary functionality, such as:

- Ability to add, edit and delete contacts
- A table index view of available contacts
- A Mini Cards index view of available contacts
- A detailed contact view listing all the specific properties of a contact
- Support a variety of pre-defined templates at contact creation
- Ability to create, edit and groups, which are collections of contacts
- View persistence (the application will remember the state of the contact views)
- Contact persistence (the application will remember the contact data in a rudimentary repository)
- Ability to create and change contact properties "on-the-fly" to demonstrate ability to alter RDF schema at runtime

6. Sample Parcels
We also include several sample parcels to demonstrate the ease of creating applications with our ViewerParcel framework even in its current infant state.

- Repository: An object inspector that provides a developer details to every information item stored in the current repository
- TimeClock: A simple utility for keeping track of billable hours.
- ZaoBao: A basic RSS aggregator

7. Roster Parcel and Sharing Views
Our roster parcel demonstrates basic instant messaging capabilities through a Jabber server. In addition, we use Jabber to demonstrate sharing of views between two Chandler clients, as long as they are subscribed on each other's Jabber rosters. View sharing is support on the Calendar, Contacts and ZaoBao parcels. You can even overlay a remote calendar view on top of your local calendar!
8. Item Repository and Strawman Schema
We've been doing a lot of design work on the Chandler Repository, which will store Items. Not all of that work will make it into the 0.1 release, we're still sorting out various repository issues. For the 0.1 Release, we have a simple item store, persisted using ZODB. We'll also have a strawman schema for Chandler, basically just a starting point:

- Items will persist in a local repository
- The chandler schema will be defined in python, which could in theory generate RDF
- We'll describe the strawman schema in RDF/XML as well
- The schema can be altered at runtime

Current Licensing Plan -- Open and Commercial Licenses

The 0.1 version of Chandler is available under the **GNU General Public License, version 2**.
We expect that subsequent versions of Chandler will also be available under one or more additional licenses. For more detail on our licensing plans, see the **Chandler Licensing Plan.**

Chandler Releases

Release 0.1 is our first public Chandler release, made available April 21st, 2003:**Chandler Release Download Page**