Planet Feed Reader: Better Living Through Gravity

Mary Gardiner, PhD student, Macquarie University[1]

email mary@puzzling.org

Abstract

This paper describes both the technical and social history of the Planet feed reader software. The Planet websites displaying recent weblog entries by members of a Free Software community are extremely popular and grew rapidly after the initial creations of Planets by very large and dynamic Free Software communities. The technical history of the software has largely been driven by the social needs of the communities, but has faced challenges common to smaller projects maintained on a part-time basis: extremely irregular releases, difficulty identifying the definitive release of the software, and quality issues.

Introduction

The Planet software—available from http://planetplanet.org/—aggregates the syndication feeds of websites and outputs them onto a single HTML page, most recent entries first. It has been widely adopted by Open Source communities to create a community "Planet" website aggregating the weblogs of the project's developers. Planets[2] fill a niche in communities between technical mailing lists, which are largely impersonal, and real-time forums like IRC which are limited to an in-group with a lot of free time. Planet websites open the social communities that surround Free Software projects to the world by allowing the participants to share both their technical thoughts and personal lives with the community via the Planet websites. It helps developers and users see members of the core community as more human.

The Planet software is written in Python. It makes heavy use of third party Python libraries, particularly the Feed Parser software available from http://feedparser.org/ and the htmltmpl library available from http://htmltmpl.sourceforge.net/. It is deliberately simple in its aims: it is configured using a single INI style text file, it stores its cached data in the filesystem and it outputs static HTML and XML files suitable for access over HTTP. It is normally run as a scheduled job (for example via cron), checking for new feeds and generating new files as determined by the maintainer's schedule.

History

The original Free Software weblog community—Advogato at http://www.advogato.org/— was created by Raph Levien in 1999, and the ability of developers with Advogato accounts to be able to make diary entries was an afterthought. Nevertheless, it was the most popular part of the site, with discussion in the diary section of the site far exceeding the articles section in quantity and often in quality too. However, as weblogging became more popular in other Internet communities and began to coalesce around self-hosted software like Movable Type and aggregation via RSS became more common, dissatisfaction with Advogato began to grow, aided by the increasingly hostile political discussion between diary entries after September 2001. Advogato remains active to the present day, but in 2002 and 2003 a number of the most prominent writers announced that they were moving to self-hosting of their diaries and ceased writing on Advogato. Interested readers

1 This is the camera-ready copy of a paper that was in the OSDC 2006 proceedings. Citation: Mary Gardiner (2006) Planet feed reader: better living through gravity in *Proceedings of the Open Source Developers Conference* (OSDC 2006), December, Melbourne Australia.

2 The frst and best known Planet websites are Planet GNOME at http://planet.gnome.org/ and Planet Debian at http://planet.debian.org/. Others are listed at http://planetplanet.org/.

presumably sought these weblogs out individually.

The first website named “Planet”—Planet GNOME, originally hosted in Jeff Waugh's personal webspace on the gnome.org servers—pre-dates the software. Waugh generated Planet files using the aggregator Spycroll: originally “Planet” was the name for the website. However, when Scott James Remnant founded Planet Debian he found the software inadequate for his purposes, and thus the fork of Spycroll known as Planet. The Planet software aimed to solve several problems that emerge from the many different errors it is possible to make when producing an XML syndication feed, with mixed success.

The problem with feeds

XML syndication feeds are a mechanism to provide the following in a machine readable format:

1. some meta-data concerning the source of the information: for example, the original URL, the author
2. a list of recently changed or added (usually added) text snippets from that data source with associated:
    - titles
    - source URLs
    - timestamps
    - unique permanent identifiers (ideally)

The text snippets are often full weblog entries (or newspaper articles or other updated source material), but may be summaries or excerpts of the text at the source URL. Sometimes only a title is provided. There are two main competing syndication formats: RSS (the acronym stands for a number of different terms: Really Simple Syndication, Rich Site Summary, RDF Site Summary)[3] and Atom[4].

The aim of Planet and its predecessor Spycroll is to take a number of XML syndication formats, extract the text snippets, and present them to a reader on a single web page. Planet, via the Feed Parser software, can process the many variants of RSS and the several versions of Atom. The single webpage has the format that most blogs have, that is, entries are provided in reverse chronological order, newest entries first, so that readers can read down until they run out of new material to read.

The single major problem with very simple aggregators like Spycroll that trust all information in XML syndication feeds is that elementary parts of that data are often wrong. For example, timestamps can be formatted in several different ways, and not all specifications explain which format to use. Many feeds do not include any timezone information, and many more do not provide correct timezone information. This provides a problem when it comes to sorting the entries in a useful format for the reader (entries they haven't read first, entries they have read second). Another piece of information that is often untrustworthy is the unique identifier, when provided. Feed authoring tools are only somewhat reliable at providing unique identifiers, frequently assigning new identifiers when the source URL schemes change, for example.

This presents a considerable problem when trying to identify entries that haven't been seen by the software before. The eventual solution that Scott James Remnant came up with was to ignore all feed provided dates, and simply assign a date based on when the Planet software first saw the entry. This involved introducing permanent storage to the software, so that for any given entry the software could look up the original date at which it was seen. Changing 'permanent' identifiers is

3 See http://www.rssboard.org/rss-specification for an example specification
4 See http://tools.ietf.org/html/rfc4287 for the specification

an unsolved problem.

Venus

By late 2005, the Planet codebase had become rather unwieldy, at least considering its size. In particular, the code was not unit tested at all, and until early 2005 there was no bugtracker. One of the major problems was that of using BSD DB for the cache of entries. The cache was fairly badly designed in the first place: it stored the full text of all entries every seen by the software, and in order to generate the Planet output it loads the entire cache into memory, leaving some users with unusable systems when they tried to generate a Planet using a cache with more than a year's worth of entries.

With a very small community, the energy available for a re-write was fairly low. However, Sam Ruby, who is involved in Feed Parser development, made an experimental re-write called Planet Venus[5] in August 2006. Venus uses plain text files, one per entry, as the cache. It includes a large number of unit tests, and shows significant improvements in memory usage.

Conclusion

The Planet revolution is mostly a product of the social impact of the various Planet webpages have had on their communities, making their personalities and diversity more obvious. The software itself still has a way to go to be solid and stable.

5 http://intertwingly.net/code/venus/