Simplicity and minimalism in software development

Introduction

- My name is Mattias Sundblad, I have been working as a software developer since 2006.
- I have worked for large corporations, small start-ups and non-profit organisations.
- First started working for a non-profit organisation in 2010. Since then, I have been both employed and worked as a volunteer at various times.
- I used to work mostly with Java, Python and Django.

Examples of projects

- Webshop, still in use.
- Previous version of the main website
- Various systems used internally

Involvement in the Picolisp community

- Involved for around 5 years
- Provided Swedish localisation, translations
- Wrote manuals and documentation
- Regular visitor in community IRC channel, #picolisp on Freenode

Picolisp?

- A language in the Lisp family of programming languages. Examples of other members: Common Lisp, Emacs Lisp, Clojure.
- Picolisp was created by Alexander Burger and has been in production use since 1988.
- Implements a Lisp language on top of a simple virtual machine architecture.
- Runs on 32- and 64- bit Linux/ BSD, Android. Intel and ARM CPU

Discerning features of Picolisp

- Simplicity.
- Expressiveness.
- A language and a complete programming system. Very few external dependencies.
- An interpreted Lisp. No compiler involved. The code you write is the code that runs.
- Efficient, both in terms of machine resources and programmer effort.

Writing Picolisp applications

- The system comes with an HTML- based GUI library. Applications are accessed through a web browser.
- Database functionality is included in the language. No need to leave Lisp to deal with PostgreSQL, MySQL etc.
- HTTP server included. No need to setup and configure Nginx, Apache etc.
- REPL included. Enables a very efficient development style
- Applications are usually very small. The code ends up in Kilobytes. More on that later.

A real life example

- A system to handle donations in testaments.
- First a Lotus Notes application, then migrated to a solution written in Python/ Django.
- When new functionality was requested, a new version was written in Picolisp.
- Now incorporated in a unified application platform.

Result of re-writing in Picolisp

- Size of Django version: 3.4 Mb, spread out over 6 directories and 68 files. Not counting external dependencies.
- Size of Picolisp version: 92 Kilobytes in 1 directory and 14 files. Provides more functionality than the old version.
- The server no longer needs Nginx, uWSGI, PostgreSQL.
- Runs on inexpensive Linux VPS with 5 Gb storage and 256 Mb RAM. Plenty of free space even with this configuration.

Current production database

- Database holds information about donors, legal contacts, payments and prospective donors.
- In total 1009 different objects.
- Size on disk: 1.1 Mb (initial version). After layout optimisation 920 Kb.
- Two nightly backup routines. One copying the entire set of database files, the second one dumps the data in CSV format and e-mails a compressed archive.
- Size of compressed archive: 42 Kb.

Benefits of minimalism

- Easy to get started, to study the problem at hand and learn things about the task while writing code.
- Users learn new things while using the applications. A Kilobyte sized program can easily be re-written to fit evolving requirements.
- Fewer dependencies. Your program is not in the hands of others through external dependencies.
- Easier, quicker and cheaper. Both when it comes to running the server, handling backups etc.

Benefits of simplicity

- Possible to understand the entire system.
- Less energy spent thinking about the tools in use, more about the actual task at hand.
- No difference between production and development environments any more.
- No context switching. Same language all the way from GUI, through app server, down to database level.
- A simple system can be used to create a solution that fits the problem. Not the other way around.

Minimalism and simplicity – why bother? It is 2018 now!

- Storage and RAM is virtually unlimited. What is the point of caring whether the application is 300 Kb or 300 Mb?
- Yes, computers can store and handle almost unlimited amounts of code today. But what about you?
- New applications can be constructed from existing parts already. Why bother writing things yourself?
- Learning and evolving. You will get better, and this is something that does not go away. It is an effort, but an effort well worth it. Besides, there really is not that much code to write in Picolisp.

Some elephants in the room

- "Lisp has all the visual appeal of oatmeal with fingernail clippings mixed in." ~ Larry Wall
- Oh really.. well, well, well --> “perl -e '$_=`perl -v`;while(/\d\,(\s\x4c[^\n]+)/){print$1}'”
- Big risk depending on technology X, where technology X == “Thing I have never heard of”
- Learning new technologies take time, and I need to solve problem Y NOW!!
- If this is such a great thing, how come it is not in use by everyone?

Thank you!

Mattias Sundblad, mattias@inogu.se