Gmail **Abel Normand <abel.ze.normand@gmail.com>**

[erlang-questions] simple virtual file system in Erlang?

Писем: 3

Marco Molteni <marco.molteni@laposte.net> 26 марта 2017 г., 17:45
Кому: Erlang-Questions Questions <erlang-questions@erlang.org>

Hello colleagues,

I would like to start from the presentation "Build an FTP Server with Ranch in 30 Minutes" and really build an FTP server, so at a certain point I need to hit the filesystem.

Since the idea is still as a presentation (as opposed to building a production FTP server) I don't want to touch the real filesystem of the host, I want to use a as simple as possible virtual filesystem. It could be backed by DETS or Mnesia for example, or it could stay only in memory.

On the other hand, it must still behave as a filesystem, that is, it must be hierarchical, so a direct mapping to a key/value store would not be enough. In the spirit of simplicity, I don't need any concept of read/write permission, I simply need a sort of graph with two types: inner nodes are directories, leaf nodes are files or empty directories.

I am thinking to use DETS and somehow introduce a very simple intermediate layer that would offer the impression to be in a graph (each non-leaf node a directory) and map it to the DETS key/value API.

Any suggestions?

thanks
marco

[1] https://ninenines.eu/articles/ranch-ftp/

_____
erlang-questions mailing list
erlang-questions@erlang.org
http://erlang.org/mailman/listinfo/erlang-questions

Jesper Louis Andersen <jesper.louis.andersen@gmail.com> 26 марта 2017 г., 19:57
Кому: Marco Molteni <marco.molteni@laposte.net>, Erlang-Questions Questions <erlang-questions@erlang.org>

The simplest way is probably to map a path to its data, where the path is a list of components. That is, the UNIX file ls in /usr/bin would be represented as ["usr", "bin", "ls"].

But you could also go the way of Plan9's venti. Map something like ["usr", "bin", "ls"] into a sha3 checksum of the data currently residing in the file and keep the data inside a storage suitable for storing data by its content addressing. venti also extends this by storing a tree of 64 kilobyte blocks which can then be regarded as the data when taken together. Thus, you have two components: the data store, and the path mapping service, mapping a path to the underlying data.

This idea has the advantage of being practically useful in many situations beside a toy example :)
[Цитируемый текст скрыт]

_____
erlang-questions mailing list
erlang-questions@erlang.org
http://erlang.org/mailman/listinfo/erlang-questions

Joe Armstrong <erlang@gmail.com> 26 марта 2017 г., 20:31
Кому: Jesper Louis Andersen <jesper.louis.andersen@gmail.com>
Копия: Erlang-Questions Questions <erlang-questions@erlang.org>

Yes - mapping the path to a content addressable store address would be great. It would also be great in a wider context.

A fun extension would be to add a an http interface. Something like

POST <blob>

To store <blob>, then later

GET /blob/sha256/<SHA>

to recover the blob. <SHA> is the SHA256 checksum of the data (the path contains the
type of the checksum - so you might say GETblob/md5/<MD5CHECKSUM>

then add a DHT - then run on every webserver on the planet
then store all data forever.

The nice thing about this is that it's self-securing - a person-in-the-middle cannot change the data without it being detected (since you can check the SHA of the
data when you get it back)

Such a system has three relatively simple layers

- transport (say HTTP over TCP)
- DHT
- storage

If you ask Jesper nicely he'll tell you all about DHT's :-)

Have fun with the storage layer for this (and no it's NOT a toy project)

/Joe
[Цитируемый текст скрыт]