OASIS-DB: a CPAN for OCaml

Building and distributing OCaml libraries and applications

Sylvain Le Gall <sylvain.le-gall@ocamlcore.com>

Presentation at OCaml Meeting 2011

April 15th, 2011

_oasis

```
OASISFormat:  0.1
Name:         ocaml-fastrandom
Version:      0.0.1
Synopsis:     Fast random number generator
Authors:      Sylvain Le Gall
License:      LGPL-2.1 with OCaml linking exception
Description:
  A random number generator compatible with standard[...]

Library fastrandom
 Path:      src
 Modules:   FastRandom
 CSources:  FastRandom_stub.c
 CCOpt:     -O2

Executable "Test"
 Path:           tests
 MainIs:         Test.ml
 Install:        false
 BuildDepends:   oUnit, fastrandom

Test test
  Command: $test
```

- Copy Cabal file format
  - Fields
  - Sections
  - Freeform
  - Conditional
- Simple text file
  - Easy to read and write
  - Beginners can understand it

How OASIS works

Beginners customize...

Advanced users customize...

_oasis

oasis setup

setup.ml

src/libfoo.clib

src/foo.mllib

ocaml setup.ml -configure

ocaml setup.ml -build

ocaml setup.ml -test

ocaml setup.ml -install

Already using OASIS

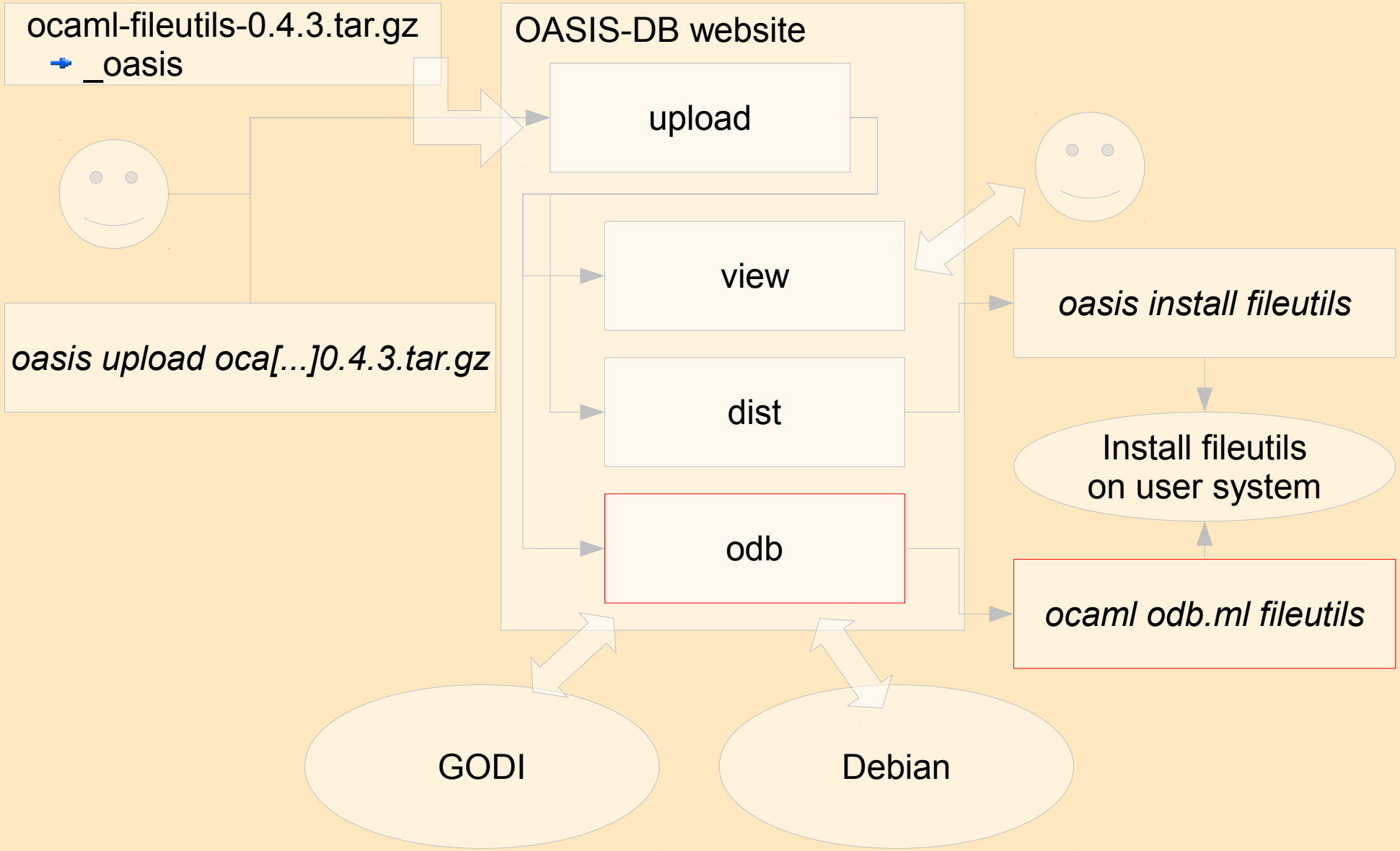
- Now
  - Cryptokit
  - Extunix
  - ocaml-text
  - OUnit
  - ocaml-sqlexpr
  - ocaml-sphinx
  - ocaml-expect
  - oasis
  - Xenops
  - Lwt
  - Sexplib
  - Jane Street Core
  - oasis-db
  - oasis2debian
  - ocaml-data-notation
  - ocamlify
  - ANSITerminal
  - bin-prot
  - type-conv

On 2011/04/11, 530 downloads
Available in GODI (release line 3.11)
Uploaded to Debian (in NEW queue)

What is CPAN/Hackage?

- Meta data about the package
  - Human readable and easy to parse
  - Dependencies to be processed automatically
  - Extra descriptions (VCS, authors, homepage)
  - This data should be also directly useful to the upstream author
- Website where upload is permitted
  - Immediate publication
  - Backup for tarballs
  - Can server as the homepage for lightweight project
- Command line to upload and download for the website
  - Allow to compose new workflow to publish
  - Ease the immediate use of uploaded packages

How OASIS-DB works



Demonstration

Future projects

- OASIS QA:
  - Integrates OASIS, OASIS-DB and oasis2debian with Debian QA tests
  - Runs our own QA tests (OASIS tests, oug or ocaml-metrics)
- oasis2rpm and godiva-oasis
  - Based on oasis2debian
  - Ease the work of packagers:
    - detection of new dependencies
    - list documentation to build
    - Standardize entry points in build system
- OASIS github integration
  - Automatically publishes tarball with _oasis from github
  - Detect new tag in git and create a tarball out of it

Future projects

- OCaml platform
  - Haskell platform is fully integrated with cabal
  - Haskell platform compared to OCaml:
    - Ulex (alex in Haskell platform)
    - OASIS install (Cabal-install)
    - Ocamlgraph (fgl)
    - Lablgtk and lablgl (GLUT, OpenGL)
    - Menhir (Happy)
    - Ocsigen (HTML, HTTP, xhtml)
    - OUnit (HUnit)
    - ocamlnet (network)
    - Kaputt (QuickCheck)
    - Pcre-ocaml (regex-*)
    - Camlzip (zlib)
    - Missing: parsec, mtl, parallel
  - Possible extensions:
    - Sexplib
    - Camlbz2
    - Extlib
    - Batteries
    - ...

For all these projects, I am looking for help

Questions ?

Extras

What are plugins?

- It translates an OASIS package data structure
- There are four kinds:
  - Conf
  - Build
  - Test
  - Doc
  - Install
  - Extra
- It can create extra fields in “_oasis”
  - “XCustomClean: $make clean”
- It can embed code into “setup.ml”

Some plugins

- None (conf, build, doc, test, install)
  - It does nothing and fail
- Custom (conf, build, doc, test, install)
  - It calls a shell command
- OCamlbuild (build)
  - It generates .mllib
  - It calls ocamlbuild with the right targets (e.g “ocamlbuild test.cma” or “ocamlbuild test.cma test.cmxa”)
- OcamlbuildDoc (doc)
  - It generates .odocl
- InternalInstall (install)
  - It installs what has been built using ocamlfind or cp
- META (extra)
  - It creates META files including build dependencies

Customization

Generated files

_tags

```
# OASIS_START
# DO NOT EDIT (digest: 1478ef[...]b2e38)
# Library odn
# Library pa_noodn
# Executable test
<tests/test.byte>: use_odn
<tests/test.byte>: pkg_str
<tests/test.byte>: pkg_oUnit
<tests/test.byte>: pkg_fileutils
<tests/*.ml>: use_odn
<tests/*.ml>: pkg_str
<tests/*.ml>: pkg_oUnit
<tests/*.ml>: pkg_fileutils
# Library pa_odn
"src": include
<src/*.ml>: pkg_type-conv
<src/*.ml>: pkg_camlp4.quotations.o
<src/*.ml>: pkg_camlp4.lib
# OASIS_STOP

<src/pa_odn.ml>: syntax_camlp4o
<src/pa_noodn.ml>: syntax_camlp4o
```

Allow to check generated content changes

Delimit insertion area for generated content

Generated content

Custom content

Variables

- The general form is “$var”
- It can be recursive:
  - $docdir
  - $datarootdir/doc/$pkg_name
  - $prefix/share/doc/ocamlify
  - /usr/local/share/doc/ocamlify
- You can use functions to transform it:
  - utoh: Unix to host for filename
  - ocaml_escaped: String.escaped
- Origin:
  - Default value
  - From file “setup.data” (static after configure step)
  - From file “setup.log” (change each time you build something)
  - From command line
  - Environment

_oasis

- Copy Cabal file format
  - Fields
  - Sections
  - Freeform
  - Conditional
- Simple text file
  - Easy to read and write
  - Beginners can understand it

```
OASISFormat: 0.1
Name:        with-c
Version:     0.0.1
Authors:     Sylvain Le Gall
LicenseFile: LICENSE
License:     LGPL with OCaml linking
  exception
Synopsis:    Minimal project with C file.
Plugins:     META

Library "with-c"
  Path: src
  Modules: A
  CSources: A_stub.c

Executable "test-with-c"
  Path: src
  MainIs: main.ml
  CompiledObject: byte
  BuildDepends: with-c
  CSources: main_stub.c
```

setup.ml

- Result of the “_oasis” compilation into a build system:
  - Standard entry points (“ocaml setup.ml -configure” or “... -build”)
  - Standalone ocaml script
  - Embed a version of _oasis already parsed (odn)
- External commands when stdlib is not enough:
  - ocamlfind
  - ocamlc -config
  - cp, rm (Sys.os_type dependent)

setup.data and setup.log

- setup.data
  - Store the result of the configure step
  - Key-value files
  - Syntax that allows its inclusion into a Makefile or an sh script

```
ocamlfind = "/usr/bin/ocamlfind"
ocamlc = "/usr/bin/ocamlc.opt"
ocamlopt = "/usr/bin/ocamlopt.opt"
ocamlbuild = "/usr/bin/ocamlbuild"
pkg_name = "oasis"
pkg_version = "0.2.0~alpha1"
```

- setup.log
  - Register events that happen after the configure
  - Helps to know what has been built and install it

```
"is_built_exec_OASIS" "true"
"built_exec_OASIS" "[...]/_build/src/cli/OASIS"
"is_built_exec_ocamlmod" "true"
"built_exec_ocamlmod" "[...]/_build/src/tools/ocamlmod"
"is_built_lib_oasis" "true"
"built_lib_oasis" "[...]/_build/src/oasis/oasis.cma"
```

OASIS-DB and GODI

- Already some packages in GODI
  - ounit
  - ocamlify
  - ocaml-expect
  - oasis (itself)
- What need to be changed in GODI
  - Allow to use other command than “make $ALL_TARGETS” to build
  - Make PLIST availables (for oasis2godi), esp. generated ones
- Mutual benefits
  - Easier upgrade of packages in GODI (easy to know what new targets, new dependencies have been added/removed)
  - Easier creation of packages in GODI (standardize entry points in the build system, parseable list of dependencies)
  - GODI is a fully fledged, well known and well established source distribution, an ideal target for OASIS-DB

OASIS-DB and Debian

- Already some packages in Debian
  - ocaml-sqlexpr
  - ocaml-expect
  - ocamlify
  - oasis
  - Ocaml-extunix
- Use already existing oasis2debian tools
  - Almost able to create a debian package out of _oasis
  - Tested on all packages using _oasis that are now in Debian
- What need to be done
  - Enhance the tool to cover more cases
  - Allow to display a clear difference when upgrading packages