Packaging Software for OS X with The Luggage

Joe Block <jpb@ooyala.com>

ooyala®

Get the code

- `git clone git://github.com/unixorn/luggage.git`

Why do you even need packaging?

- Convenience
- Repeatability
- Less user aggravation
- System Management Software (Puppet, Chef, Casper, ARD)
- InstaDMG
- Munki

o·o·y·a·l·a®

Alternatives

- Apple's PackageMaker - Buggy, GUI makes it hard to get peer review
- JAMF Composer - GUI, so still hard to get peer review, though it works well. Totally worth buying just for dealing with Creative Suite.
- The Luggage

Benefits of The Luggage

- Text-based configuration
  - Makefiles are text, so you can check them into source control and get a meaningful revision history later.
  - You can get coworker peer-review of your changes easily, since the diffs are human readable.
  - Packages are made with Makefiles, so you can find references online when you need to customize your package.

Bootstrapping The Luggage

- `git clone git://github.com/unixorn/luggage.git`
- `cd luggage`
- `sudo make pkg`
- `sudo installer -target / -pkg ./luggage-YYYYMMDD.pkg`
- or `sudo make bootstrap_files`

Basic Luggage Commands

- Deployment
  - **`make dmg`** - Build the package and wrap it in a compressed dmg.
- Development
  - **`make pkgls`** - Build the package, then print an ls-style listing of the contents so you can confirm that it looks correct.

Breaking down a Luggage Makefile

ooyala

Mandatory entries

- `include /usr/local/share/luggage/luggage.make`
- TITLE
- REVERSE_DOMAIN
- PAYLOAD

Set TITLE

```
# Example of packaging a gui application
#
include /usr/local/share/luggage/luggage.make

TITLE=example-firefox
REVERSE_DOMAIN=com.example.corp
TARBALL_LOCATION="http://localhost/tarballs/firefox-2010030901.tar.bz2"
PAYLOAD=unbz2-applications-Firefox.app

# Having the tarball depend on Makefile ensures that make automatically
# downloads a new tarball when you update TARBALL_LOCATION, rather than
# building a stale payload.

Firefox.app.tar.bz2: Makefile
    curl ${TARBALL_LOCATION} -o Firefox.app.tar.bz2
```

Set REVERSE_DOMAIN

```
# Example of packaging a gui application
#
include /usr/local/share/luggage/luggage.make

TITLE=example-firefox
REVERSE_DOMAIN=com.example.corp
TARBALL_LOCATION="http://localhost/tarballs/firefox-2010030901.tar.bz2"
PAYLOAD=unbz2-applications-Firefox.app

# Having the tarball depend on Makefile ensures that make automatically
# downloads a new tarball when you update TARBALL_LOCATION, rather than
# building a stale payload.

Firefox.app.tar.bz2: Makefile
    curl ${TARBALL_LOCATION} -o Firefox.app.tar.bz2
```

Set REVERSE_DOMAIN

```
# Example of packaging a gui application
#
include /usr/local/share/luggage/luggage.make

TITLE=example-firefox
REVERSE_DOMAIN=com.example.corp
TARBALL_LOCATION="http://localhost/tarballs/firefox-2010030901.tar.bz2"
PAYLOAD=unbz2-applications-Firefox.app

# Having the tarball depend on Makefile ensures that make automatically
# downloads a new tarball when you update TARBALL_LOCATION, rather than
# building a stale payload.

Firefox.app.tar.bz2: Makefile
    curl ${TARBALL_LOCATION} -o Firefox.app.tar.bz2
```

Optional settings

- PACKAGE_VERSION

Set PACKAGE_VERSION (Optional)

```
# Example of packaging a gui application
#
include /usr/local/share/luggage/luggage.make

TITLE=example-firefox
REVERSE_DOMAIN=com.example.corp
TARBALL_LOCATION="http://localhost/tarballs/firefox-2010030901.tar.bz2"
PAYLOAD=unbz2-applications-Firefox.app
PACKAGE_VERSION=2010110101

# Having the tarball depend on Makefile ensures that make automatically
# downloads a new tarball when you update TARBALL_LOCATION, rather than
# building a stale payload.

Firefox.app.tar.bz2: Makefile
	curl ${TARBALL_LOCATION} -o Firefox.app.tar.bz2
```

o·o·y·a·l·a®

Configuration Examples

Installing GUI app (Firefox)

```
# Example of packaging a gui application
#
include /usr/local/share/luggage/luggage.make

TITLE=example-firefox
REVERSE_DOMAIN=com.example.corp
TARBALL_LOCATION="http://localhost/tarballs/firefox-2010030901.tar.bz2"
PAYLOAD=unbz2-applications-Firefox.app
PACKAGE_VERSION=2010030901

# Having the tarball depend on Makefile ensures that make automatically
# downloads a new tarball when you update TARBALL_LOCATION, rather than
# building a stale payload.

Firefox.app.tar.bz2: Makefile
	curl ${TARBALL_LOCATION} -o Firefox.app.tar.bz2
```

Install a GUI app + scripts

```
include /usr/local/share/luggage/luggage.make

TITLE=enable_loginhooks
REVERSE_DOMAIN=com.example.corp
PAYLOAD=\
    unbz2-utilities-iHook.app pack-hook-background.jpg \
    pack-hookscript-L0999_DestroyKerberosTicket.hook \
    pack-hookscript-login.hook pack-hookscript-logout.hook \
    pack-hookscript-run_loginscripts.rb pack-hookscript-run_logoutscripts.rb \
    pack-hookscript-hookutils.rb pack-script-postflight

TARBALL_LOCATION="http://localhost/tarballs/iHook.app.20091009.tar.bz2"

# stick a background image into /etc/hooks to use with iHook.
pack-hook-background.jpg: hook_background.jpg l_etc_hooks
    @sudo ${INSTALL} -m 644 hook_background.jpg ${WORK_D}/etc/hooks/hook_background.jpg

# local copy depends on Makefile to prevent stale payloads
iHook.app.tar.bz2: Makefile
    curl ${TARBALL_LOCATION} -o iHook.app.tar.bz2
```

o·o·y·a·l·a®

Common Pain Points

- GUI Apps with spaces in their names
- Makefile mysteriously breaks with a “missing separator” error
- Builds work on my workstation, but the same exact Makefile fails when I build on a file share

Dealing with a space in the App name

```
include /usr/local/share/luggage/luggage.make

TITLE=the_unarchiver
REVERSE_DOMAIN=com.example.corp
PAYLOAD=utility-the_unarchiver
PACKAGE_VERSION=2009121001

TARBALL_LOCATION="http://localhost/tarballs/the_unarchiver_2.2.tar.bz2"

the_unarchiver.app.tar.bz2: Makefile
	curl ${TARBALL_LOCATION} -o the_unarchiver.app.tar.bz2

utility-the_unarchiver: the_unarchiver.app.tar.bz2 l_Applications_Utilities
	@sudo ${TAR} xjf the_unarchiver.app.tar.bz2 -C ${WORK_D}/Applications/Utilities
	@sudo chown -R root:admin "${WORK_D}/Applications/Utilities/The Unarchiver.app"
```

o·o·y·a·l·a®

“missing separator” error

You’ve got leading spaces in one of your rule stanzas instead of a tab. Make cares (a lot) about the difference.

This is the most common problems people unfamiliar with make run into.

Makefile works on workstation, but breaks when I build on a server directory

Don't do that. Build on a local HFS+ filesystem, or you're going to have problems.

o·o·y·a·l·a®