Top Tools for Admins

MacRetreats 2005

Scott M. Neal
Senseption

Goal of Session

- Empower non-programmers to utilize developer tools to solve problems
  - Develop a clear understanding of OS X's primary configuration file format: Property Lists
  - Utilize existing command-line tools with a GUI that YOU develop
  - Automate repeated tasks that would normally be boring and prone to error
  - More spare time for you!
- This session is designed with walk-through examples for you to follow
  - Hopefully you've already installed the developer tools!

Developer Tools will Improve Our Lives

- Problems we will solve today with Developer Tools
  - Property Lists and User Defaults editing
  - Creating Repeated tasks using `launchd`
  - Prettying-up shell scripts and commands by adding a GUI
  - Automating menial tasks

Developing on OS X

- Cocoa
  - Interface Builder
- Carbon
  - Interface Builder
- WebObjects
  - EOModeler
  - WebObjects Builder
- Java
- BSD-compliant UNIX code
  - command line compilers
  - X11

Developing on OS X: Tools in this Session

- Applescript
  - Script Editor
- Applescript Studio
  - Interface Builder
- Automator
  - Automator (from the department of redundancy department)

XCode

- Development environment provided with OS X Developer Tools
- Provides templates for projects
  - Initial configuration files
  - Organized structure that is project type-specific
  - VERY configurable

Property Lists & Defaults

- Configuration information for processes needs to be stored somewhere
- Apple developed the Property List (aka plist) Format to provide a consistent way to store information
  - Not ALL configuration information is stored in plist format
- Property Lists provide the backbone for the Defaults Preference infrastructure

Property Lists & Defaults

- Keys are specific to each process and define configuration options for that process, such as
  - Default new window location and size
  - File location/path
  - *the sky is the limit...*
- Modifying Property Lists and the Defaults system gives YOU control over the operation of processes
- Great for troubleshooting/debugging
  - "Trashing the preferences" (which really should be "Renaming the preferences")

Property Lists

- Property Lists are Key-Value storage mechanisms
  - If a process needs to store a value for a default, it provides a key for identification and a value for that key
- Values can belong to different Value Classes
  - String
  - Number
  - Boolean
  - Date
  - Array
  - Dictionary
  - Data

Property List Formats

- There are currently 3 Property List file formats:
  - XML
  - ASCII (NeXT-style)
  - Binary
- Property Lists are stored with the file extension .plist
- The Property list infrastructure can read/write all 3 formats
  - Humans can read/write the first 2 formats
  - Binary format human-unreadable, but is parsed MUCH faster by the OS
    - faster booting
    - faster application launching

Property List Formats: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>AppendAMPM</key>
    <true/>
    <key>ClockDigital</key>
    <integer>1</integer>
    <key>ClockEnabled</key>
    <true/>
    <key>ClockLocation</key>
    <integer>0</integer>
    <key>DisplaySeconds</key>
    <false/>
    <key>FlashSeparators</key>
    <false/>
    <key>LastSavedGlobalTimeString</key>
    <string>h:mm:ss a</string>
    <key>PreferencesVersion</key>
    <integer>2</integer>
    <key>ShowDay</key>
    <true/>
    <key>Transparency</key>
    <real>0.80000001192092896</real>
    <key>Use24HourClock</key>
    <false/>
</dict>
</plist>
```

Property List Formats: ASCII/NeXT

```
{
    AppendAMPM = 1;
    ClockDigital = 1;
    ClockEnabled = 1;
    ClockLocation = 0;
    DisplaySeconds = 0;
    FlashSeparators = 0;
    LastSavedGlobalTimeString = "h:mm:ss a";
    PreferencesVersion = 2;
    ShowDay = 1;
    Transparency = 0.800000011920929;
    Use24HourClock = 0;
}
```

Property List Formats: Binary

bplist00?

FlashSeparators_PreferencesVersionWShowDay\ClockDigitalZAppendAMPM\ClockEnabled^DisplaySeconds_LastSavedGlobalTimeString]ClockLocation\Transparency^Use24HourClock Yh:mm:ss a#?�N[fs???????????

Editing Property Lists

- Text Editor/command-line text editors
  - Doesn't work with Binary (unless converted, which we will soon see how to do)
- Property List Editor
  - Located in /Developer/Applications/Utilities
  - NOT installed by default on OS X or OS X Server
    - Really should be...
    - Until it is, good thing we're here now!

Converting from one plist format to another

- "Save To" option of Property List Editor
- command-line `plutil` tool

Property Lists: Summary

- YOU are now empowered to edit property list files directly
  - double-edged sword: you can also SCREW UP a process' configuration, making your process (or even entire machine) unusable
  - Comes in handy for troubleshooting, especially in single user mode
- Any Questions?

Property Lists Locations

- OS X has a standard hierarchical resource search policy (for Fonts, etc.):
  ~/Library/*
  /Library/*
  /Network/Library/* (if it exists)
  /System/Library/*
- Property Lists are NOT searched hierarchically and can be stored ANYWHERE, including:
  - /etc/*
  - Application Bundles
  - Preferences (but where?)

Preferences: Defaults Infrastructure

- OS X has a mechanism called Defaults that is a "portal" into the Preferences plist system for processes
  - It does NOT incorporate EVERY plist on the system (phew!), only the ones dedicated to preferences
  - Processes themselves don't need to understand how to read/write or find plist files, they use the Defaults infrastructure
  - Your scripts can leverage this infrastucture and be able to save their own preferences!

Defaults Infrastructure

- Location of Preference files:
  - ~/Library/Preferences
  - /Library/Preferences
  - /var/root/Library/Preferences
- Most files in these folders are in plist format
- Note that some files in */Preferences are NOT plist format
  - These will not be accessible through the Defaults system, but are read directly by the process that owns that preference

Defaults Domains

- Preferences in the Defaults system are organized by domain (and, optionally, host)
  - Typically correspond to individual applications/processes
- Nomenclature for these domains typically (not always) follows Java reverse-FQDN syntax WITHOUT the .plist extension
  - Prevents namespace collisions
  - Examples:
    com.apple.menubar.clock
    loginwindow

Accessing Defaults

- Defaults are accessed using the CLI tool `defaults`
  - When only a Domain is specified, `defaults` searches ~/Library/Preferences for a plist file matching the Domain argument with .plist appended
- Example:
  `defaults read com.apple.menubar.clock`
- Note that output from `defaults` is in ASCII/NeXT format, independent of the plist format itself (Binary, XML, or ASCII/NeXT)
- VERY useful in automation and scripting!

Reading/Writing Values for Specific Keys

- A specific key can be read using the same `defaults` tool
  - Example:

```
defaults read com.apple.menubar.clock
ClockEnabled
```

- A key's value can be modified:
  - Example:

```
defaults write com.apple.menubar.clock
ClockEnabled 0
```

Other Defaults Domains

- `NSGlobalDomain`
  - Used for default shared key-value combos, not process or domain-specific
  - Can use `-g` instead of specifying `NSGlobalDomain`
  - Example:
    ```
    defaults read NSGlobalDomain
    AppleMiniaturizeOnDoubleClick
    defaults read -g
    AppleMiniaturizeOnDoubleClick
    ```

Other Defaults Domains (cont.)

- A full path to a .plist file (minus the .plist extension) can be specified as a domain
  - Example:
```
defaults read /Library/Preferences/
.GlobalPreferences
```
- Take a look at the Domains available in each of the Preference file locations:
  - ~/Library/Preferences
  - /Library/Preferences
  - /var/root/Library/Preferences

Specifying a Host with Defaults

- The `ByHost` folder seen in some of the previous Preference locations stores host-specific information
  - Useful for processes that utilize more than one host, and want to have host-specific preferences
- You may specify a hostname as either a MAC (IP) address or `-currentHost`
- Examples:

```
defaults -currentHost read
com.apple.networkConnect

defaults -host 000a95a92943 read
```

Where is the .plist for NSGlobalDomain?

```
~/Library/Preferences/
ByHost.GlobalPreferences.MacAddress
.plist
```
- This is an FYI, since you really should be using `defaults` to edit the global data (it's MUCH easier!)

Defaults (Summary)

- Preferences are managed through the Defaults infrastructure, and stored in plist files in specific Domains
- You can leverage the `defaults` command to read/write pre-existing domains
- You can also use `defaults` to create/read/write/delete preferences on-the-fly
  - Even scripts you write or leverage can have saved preferences--sure comes in handy for automating workflows!
- Any questions?

launchd

- The `launchd` daemon process is new to OS X Tiger, and replaces MANY previous processes and methods that controlled the launch of processes:
  - `rc` files
  - `init`
  - `inetd/xinetd`
  - `SystemStarter`
- `launchd` is controlled via Property Lists and commands that load/unload those plists

launchd.plist

- The plist files that control `launchd` are stored in one of the following places:
  - `~/Library/Launch{Agents,Daemons}`
  - `/Library/Launch{Agents,Daemons}`
  - `/System/Library/Launch{Agents,Daemons}`
- Like any plist, the files consists of a set of Keys-Values that define properties
  - You set the right keys to the right values, and YOU become the boss!

launchd.plist
Mandatory Keys

- `Label`
  - A unique identifier for this process
- `Program`
  - Absolute path to the executable file which launches the process
  - optional if `ProgramArguments` specifies launch path
- `ProgramArguments`
  - An array of arguments for the process

`launchd.plist`
Some Optional Keys

- `UserName`
  - The username under which the process should run
- `RootDirectory`
  - A directory that will be the root directory for the process (using `chroot`)
- `WatchPaths`
  - A list of paths that, when modified, will cause the process to launch (if not already launched)
- `StartInterval`
  - An interval in seconds specifying how often the process should be started
- `StandardCalendarInterval`
  - A specification of a repeated calendar interval

launchctl

- Used to load/unload launchd.plist files into the launchd system
- Must be run as root (usually with `sudo`)
- Example:

```
sudo launchctl load ~/Library/
LaunchAgents/com.apple.TextEdit.plist

sudo launchctl unload ~/Library/
LaunchAgents/com.apple.TextEdit.plist
```

Walk-Through: TextEdit

- We are going to create a simple plist file for TextEdit to ensure that it is always running

launchd (Summary)

- We have only scratched the surface of `launchd` here
  - Please consult online documentation and the man pages for `launchd`, `launchd.plist`, `launchctl`, etc.
- Any Questions?

AppleScript

- One of the (if not THE) easiest scripting languages to learn
- Syntax is flexible, and punctuation is not as necessary (semi-colons and braces and back ticks OH MY!)
- AppleScript can easily call Unix scripts or programs (this bullet is what your English teacher would call "foreshadowing")
- Unix scripts and programs can easily call AppleScripts through the Open Scripting Architecture (OSA)

Apple Events

- Apple Events are commands that are sent to an application
- MANY applications AppleScript-able through Apple Events
- Apple Events are defined for each Apple Event-compatible application
  - Standard Suite
  - Application-specific
- Use Script Editor to see what Apple Events an application supports

AppleScript Example

```
tell application "Finder"
  set finderWins to (every Finder window)
  repeat with w in finderWins
    set finderWindowName to name of w
    display dialog finderWindowName
  end repeat
end tell
```

ScriptEditor Record Mode

- Will record your actions and create AppleScript commands from them!
- Great for creating a template script that you can modify/addend
- Limited by what events the application provides for recording
  - It often seems like the ONE THING you need is missing

Bringing Executables into AppleScript

- So how can we incorporate programs that don't respond to Apple Events into our flow?
- How can we make applications that only run in the Terminal (referred to as Tools by Apple, rather than Applications) be double-clickable for neophyte, Terminal-phobic users?

AppleScript Example

```
property command : "ls   "

set x to display dialog "Enter path to
show listing" default answer "" buttons
{"Cancel", "Show Me!"} default button
"Show Me!"
if button returned of x is "Show Me!" then
  try
    set filePath to text returned of x
    set output to do shell script command
    & filePath
    display dialog output
  end try
end if
```

Running AppleScripts from Unix scripts

- Use `osascript` command
- OSA stands for Open Scripting Architecture
  - There is an OSA-compatible implementation of JavaScript
    http://www.latenightsw.com/freeware/JavaScriptOSA/
- So why is this cool...?
  - Answer: UNIX scripts can now ALSO access AppleEvent-supported applications
    - Quicktime compression
    - Adobe CS
    - etc.

AppleScript Droplets

- AppleScripts can be written as Droplets, which allow things to be "dropped" onto them
- Dialogs can then be optional, and the script will operate on all files/folders dropped onto it
- Great for workflow automation

AppleScript Studio: Adding a GUI

- We have seen how to make AppleScripts communicate with pre-existing programs
- Wouldn't it be COOL if we could use Apple Developer Tools to put a fancy GUI in front of command-line tools?
- We can leverage the same Interface Builder that is used to develop Cocoa apps for AppleScripts

AppleScript & A.S. Studio Summary

- Intuitive, syntax-friendly scripting language
- Part of DNA of OS X
  - Apple Events
  - Droplets
- AppleScript Studio bring power of full-powered GUI to your scripts without ANY programming (just connect-the-dots)
- Any Questions?

Automator

- Brand-new to Tiger
- Connects applications together in a workflow without ANY scripting/programming
  - Notice the icon, and how it is "pipe-ing" :-)

Automator

- Automator Actions not directly correlated with Apple Events
  - Automator actions are added by "real" programmers to the applications so that other can use them (as are Apple Events, but in a different way)
    - Automator Actions can be written as Apple Scripts
  - Only a small subset of Apple Events have made it to Automator Actions right now (but this will change)
  - In Automator's current defense, there are some things in Automator that would be a pain to do in AppleScript

Automator (summary)

- Automator is destined to be a VERY important part of OS X
  - Still in embryonic/newborn stage right now
  - VERY good support infrastructure exists already in the community
  - If you find Automator limiting right now, be patient, this is going to be HUGE
- Any Questions?

Synopsis

- Even non-programmers have the power to harness developer tools
- You can use Property List Editor to
  - Edit process' properties
  - Control the launching of applications with `launchd`
- You can use the Preference Defaults system for troubleshooting and control
- You can create scripts/programs (or get them off the Internet!) and add fancy GUIs to them with AppleScript Studio
- You can automate workflows with Automator

Resources

- Install the Developer tools!
  - Tiger DVD
  - Download: http://developer.apple.com/
    - Get ADC membership (Online membership is free)
- Property List Editor and the Defaults Preferences System
  - man pages
- launchd
  - AFP548: http://www.afp548.com/article.php?story=20050620071558293
  - man pages
  - http://developer.apple.com/macosx/launchd.html
  - Peachpit Unix book: Unix for Mac OS X 10.4 Tiger: Visual QuickPro Guide, 2nd Edition

Resources

- Applescript and Applescript Studio
  - /Developer/ADC Reference Library/ documentation/AppleScript/Conceptual/*
  - http://www.apple.com/applescript/resources/
  - http://www.apple.com/applescript/studio/
  - O'Reilly Books
- Automator
  - http://www.apple.com/automator
  - http://www.automatorworld.com
  - http://www.automator.us/resources.html

Thank You!

Top Tools for Admins

MacRetreats 2005

Scott M. Neal
Senseption