Xcode Tiefergelegt: Effizienz und Automatisierung

oder "Wie werde ich nicht wahnsinnig?"

Navigation durch Projekte

- Befehl/Wahl-Doppelklick + Zurück-Pfeil
- Open Quickly mit Eingabe oder Auswahl
- Klammern Doppelklicken
- Zeilen Dreifachklicken, Drag & Drop (Wahltaste!)
- Spaltenweises Markieren (Wahltaste), Edit All in Scope

Navigation durch Projekte

- “Products”-Gruppe, “Reveal in Finder”
- “String Matching”-Filterfeld, “Reveal in Group Tree”
- Split View, “Grouped/Ungrouped”, “Open in Separate Editor”
- “Counterpart”, Methoden/Funktionen-Popup, #pragma mark - etc.
- Lesezeichen

Das Suchfenster

- Mehrere Dateien Durchsuchen, Index (“wer ruft das auf?”)
- Reguläre Ausdrücke + Suchen und Ersetzen
- Frühere Ergebnisse/Werte im Combo-Box-Menü
- Suchfeld in Dateifenstern hat Lupen-Popup
- Suche nach +(.*)myMethod oder ::MyMethod, Befehl-Umschalt-G für Rückwärts

Die Hierarchie

- Projekt ➙ Target
- Jeder Target kann mit Build-Konfiguration verändert werden
- Jede Konfiguration kann auf einer .xcconfig basieren (Copy/Paste, Mergen)
- Executables - Environment, Parameter & selber erstellen für Plug-ins
- Dateien relativ zu Gruppen, Projekt o.Ä.

Makros in Quellcode

- Sources und plists, oder explicit mit cpp-Tool
- -C Option gegen Kommentare in URLs, Other Preprocessor Flags
- Preprocessor Macros: DEBUG=1
- __FILE__, __LINE__, __DATE__ (nur bei neukompilieren)

Xcode User Scripts

- Eingebaut: “Insert Text Macro”-Menü: Klammern etc. (.xctxtmacro)
- Kurzbefehle zuweisen in Xcode Präferenzen
- Code generieren mit User Scripts (bash, php, Shebang hilft)
- Integrieren von Tools (z.B. häufige git-Befehle)
- <#platzhaltername#> für Platzhalter

Shell-Skripte für Verlässlichkeit

- Reproduzierbarer release-Build mit xcodebuild-Tool
- zip oder hdiutil zum Verpacken
- curl --upload

Shell Script Build-Phasen

- Eingabe/Ausgabedateien, generieren von Code vor Kompilierung
- Build-Nummer (git, svn) einfügen
- Hinzufügen von Resourcen basierend auf Flags
  - rsync vs. cp für Skripte

Shell Script Build-Phasen

- HTML-Hilfedateien mit PHP, Rails etc., Apple Help Indexing Tool
- Entfernen von .svn, .git, headers etc. aus dem Project
- Aufrufen anderer Programme (Generieren lokalisierter Grafiken...)

Vorlagen zum Abreißen

- ~/Library/Application Support/Developer/Shared/Xcode/Project Templates/MeineKategorie oder …/File Templates/…
- Beispiele in /Developer/Library/Xcode oder /Developer/Platforms/…
- Platzhalter, Plists mit extra Info. Vorsicht mit Encodings! Texteditor benutzen!

Xcode-Projekte sind lesbar

- NeXT-style Keyed Archives in .pbproj.
- Dummy-Klassen, die die interessanten Felder auslesen
- Tools, die Dependencies prüfen, Source-Pakete schnüren etc.
- Mit Vorsicht: Regex-Suchen/Ersetzen für Warnings-Flags, Pfadänderungen etc. Inspektor ist auch ganz gut (Bef.-Wahl-I)

Kleinkram

- Per-Datei GCC-Flags (Inspektor) & Per-Architektur Flags
- Go Next Error/Warning-Menüpunkt (Bef-+/=)
- Documentation Sets selbermachen (Ordner, signiertes XAR, RSS)
- Verborgene Einstellungen:
  - PBXBuildFailureSound / PBXBuildSuccessSound
  - XCCodeSenseFormattingOptions

Weiterführende Infos

- http://www.turkeysheartrhinos.com/?p=8
- http://zathras.de/blog-helpful-xcode-user-scripts.htm
- http://zathras.de/x2005-04-18.htm
- http://zathras.de/blog-xcode-regular-expressions.htm
- http://zathras.de/blog-shared-precompiled-headers.htm
- http://zathras.de/blog-the-dangers-of-zerolink.htm
- http://zathras.de/x2004-12-05b.htm
- http://zathras.de/blog-ten-two-through-ten-four-intel-compatibility.htm
- http://zathras.de/blog-objc-exception-throw-breakpoint.htm
- http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/Documentation_Sets/000-Introduction/introduction.html & http://developer.apple.com/tools/creatingdocsetswithdoxygen.html
- http://github.com/jollyjinx/jnxfree/

Danke!

Uli Kusterer
http://www.zathras.de