

Macoun

Cross Platform Code-Sharing Praxistipps

Uli Kusterer

Ablauf






- Definition
- Warum/Warum Nicht?
- Cross-Platform Frameworks
- Praxistipps

"Cross Platform"

- tvOS/watchOS/iphoneOS?
- Windows Phone/Windows 10/Xbox?
- macOS/iOS?
- macOS/Windows/Linux?

Warum Cross Platform Code?

Warum Cross Platform Code?

- Geld Sparen 
- Kunden auf mehreren Plattformen 
- Laden verkauft nicht, wenn kein Windows dabei ist 
- Weniger Entwickler nötig 
- App hat keine native UI 

Cross-Platform Frameworks

- Qt
- Andere Cross-Platform-Frameworks
- Selbergebaut

Warum nicht Qt?

- Menüs, Rollbalken
- Speichermanagement
- Hürde vor Plattform-Features
- Keine Unterstützung für Datei-Assoziationen
- Verbirgt Plattform-Unterschiede, bis sie zum Problem werden

Warum nicht andere?

- Weil sie noch schlechter sind

**"Kleinstes Gemeinsames
Vielfaches"-Design**

Warum selber bauen?

- Keine Kompromisse
- Minimal
- Selbe Menge an Entwicklern nötig
- Verwenden von Plattform-Bibliotheken, Tools und Features
- Abstraktion passt

Menschen

Das Team

- Für jede Plattform ein Entwickler
- Koordination ist das Wichtigste
 - Persönliche Treffen!
- Keiner sollte voraus rennen

Der Code

- #ifdefs vermeiden
- Plattform-Differenzen heraus-abstrahieren
 - PImpl und Unterklassen
 - Virtuelle Methoden

Der Code

- Dünne Schicht um die Plattform
- Native UI
 - Model/Model-seitige Controller cross platform
 - Views/View Controller nativ, aber cross-platform Helfer
 - AutoLayout(-wrapper) für generierte UI-Teile

Tooling

- Nav/Code Completion/Dateisystem-Brecher vermeiden
- Exotische Tools/Sprachen vermeiden
- Warnungen als Fehler (unbedingt mit CI!)

Fehlerbehandlung

- Fehler wie NSError. *nie* Fehlercodes "wiederverwenden"
- Plattform-Fehler sollten immer behalten werden
- Klar markieren, wo Exceptions kommen können

Thread-Safety

- Was bedeutet "thread safe" überhaupt?
- "Main Thread"? Haben alle Plattformen das?
- Funktionen markieren

Benennung

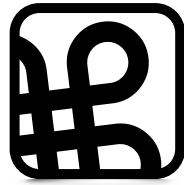
- Apple hat einen Standard
- Microsoft hat einen Standard
- Im Zweifel Kollisionen vermeiden/**definieren**

Datentypen

- Cross-Platform-Typen auf native typedefs
- Aber nur wo's 100% passt
- Datentypen markieren wenn sie persistiert werden (auch enums!)

Fragen?

Vielen Dank



Macoun