Slide 4

Es gibt einen wichtigen Unterschied, welcher nicht gleich ersichtlich ist.
Es gibt den AppStore für das iOS, in welchem Apps für iPhone, iPod und iPad angeboten werden.
Und es gibt den AppStore mit Apps für das MacOS. Und genau um diesen geht es hier.

Slide 5 – Developer.apple.com

Eine der zwei Grundvoraussetzungen um Apps in den AppStore stellen zu können, ist über ein Entwickler Konto bei Apple zu verfügen. Ein solches ist innert wenigen Minuten mit Hilfe einer Kreditkarte eröffnet und kostet ca. 90 US Dollar.

Die zweite Voraussetzung ist diejenige, dass Xcode auf dem Rechner installiert ist.
Xcode kann gratis von developer.apple.com heruntergeladen werden.

Nachdem das Entwicklerkonto eröffnet ist, müssen auf developer.apple.com zwei Zertifikate heruntergeladen werden. Es gibt je ein Zertifikat für das Signieren der App und ein Zertifikat wird für das Siginieren des Installationspaketes verwendet.

Als letzte Arbeit auf developer.apple.com muss für jede App die man in den Store stellen will eine sogenannte BundleID erstellen. Apple empfiehlt diese aus der Topdomäne, Domäne und Subdomäne zu erstellen.

Demo ()

Slide 6 - iTunesConnect

Gehen Sie nun zu itunesconnect.apple.com.
iTunes Connect ist die Schaltzentrale für Ihre Apps. Hier werden Apps angekündigt, die Preise festgelegt und vieles mehr.

Bevor Sie mit dem Erstellen einer App beginnen, überlegen Sie sich gut welchen Namen die App haben soll. Nachdem ein Appname mal angekündigt worden ist, ist der Appname für mindestens 6 Monate im System gesperrt und kann auch von Ihnen selbst nicht mehr verwendet werden.

Die SKU Nummer ist eine eindeutige Nummer für die App in Ihrem App Portfolio. Z.B. 01

Wählen Sie die korrekte BundleID aus dem Aufklappmenü aus. Sollte die BundleID hier nicht erscheinen wurde diese auf developer.apple.com noch nicht registriert.

Ok, somit sind die Vorbereitungen fürs Erste abgeschlossen.

Demo ()

Slide 7 – Wer war zuerst: Die Henne oder das Ei?

FileMaker Runtime Lösung bestehen in der Regel aus mindestens drei zwei Dateien: Der Runtime.App und der FileMaker Datenbank die dazugehört. Apple erwartet aber, dass Apps im Appstore, „Bundles“ sind, also „eine Ein-Datei-Lösung“.

Deswegen muss hier ein Trick angewendet werden: Die Runtime.app wird in eine AppleScript Hülle gepasst. Das sieht dann Schematisch dargestellt so aus:

Die AppleScript Hülle umschliesst alles. In der Hülle liegt die Runtime.app, wie auch die Signatur. Und nach einem ersten Start liegen die FileMaker Datenbanken ausserhalb der App.

Slide 8 – Appverzeichnis

Bevor ich Ihnen meinen Lösungsansatz des vorherigen geschilderten Problems zeige, möchte ich Ihnen kurz den Aufbau des App Verzeichnisses zeigen und erklären.

Demo (App Aufbau)

Dies ist der Aufbau des App Verzeichnisses wenn die App fertig erstellt und vom AppStore heruntergeladen worden ist.
Rot umrandet die Verzeichnisse die beim Signieren und beim Kauf der App erzeugt werden.
Blau umrandet ist das AppleScript.app welches als Hilfsapp dient um die FileMaker Runtime in ein programmpaket zu bringen.
Grün umrandet ist die Runtime Lösung mit dem ganz normalen Aufbau.

Slide 9 - Runtime vorbereiten

Apple hat einen ganzen Katalog von Richtlinien erstellt, die Apps im Appstore betreffen. Und wie Apple selbst schreibt ist diese Arbeit noch nicht abgeschlossen und kann jederzeit erweitert und geändert werden.

Die wichtigste Regel für FileMaker Runtime Lösungen ist, dass die App absolut keine Daten in das App Verzeichnis schreiben darf. Apple empfiehlt hierzu die Daten in das Verzeichnis „Application Support“ im Anwenderverzeichnis zu schreiben.

Bei FileMaker Runtimes muss sich die Primärdatei jedoch im gleichen Verzeichnis wie die Runtime.app befinden.

-> Demo (Starterdatei)

Wie Sie vorhin schon gesehen haben befindet sind im App selbst eine einzige FileMaker Datei. Die Primärdatei der FileMaker Runtime Lösung.
Die Starterdatei der FileMaker Runtime Lösung verfügt über ein einziges Script und wenige Felder.

-> Datenbankdefinieren öffnen
Je ein Feld vom Typ „Container“ – für jede Lösungsdatei je ein Feld.
Ein Textfeld mit globaler Speicherung – zum Einlesen der Preferencesdatei. Die globale Speicherung hier ist wichtig, da globale Textfelder auch in einer schreibgeschützten FileMaker Datei geändert werden können.

-> Dateioptionen öffnen

Das Script wird beim Öffnen der FileMaker Datei automatisch ausgeführt.

-> Scriptmaker öffnen
Nun zum Script selbst. Diese Script hier macht im groben folgendes:
-> Script erklären

->Entwicklerwerkzeuge öffnen
Mit Hilfe des FileMaker Advanced Entwicklerwerkzeuges erstellen Sie nun die Runtime. Dabei wird die Startdatei als Primärdatei angegeben.

Slide 10 – AppleScript erstellen

Nachdem nun die FileMaker Runtime Lösung vorbereitet ist, das AppleScript.app erstellt werden, welches später FileMaker Runtime Lösung aufnimmt.

Dieser Schritt wird im Artikel von Markus Schall gut erklärt.

Slide 11 – Icon Erstellen

Nun kommt die Chance ein wenig kreativ zu werden.
Erstellen Sie eine Grafik mit der Grösse von 512 x 512 Pixel. Mit Hilfe des Iconcomposers (teil vom XCode Installationspaketes) erstellen Sie nun eine Icon.icns Datei. Machen Sie 2 Kopien mit den Namen: applet.icns und eine mit dem Name FMapp.icns.

Ersetzen Sie die beiden gleichnamigen Dateien im Runtime.app und im Applescript.app mit den beiden obigen Dateien.
->Demo: Apppakete öffnen und zeigen.

Slide 12 – Info.plist

Öffnen Sie die Info.plist des AppleScript.apps und fügen Sie die neuen Einträge hinzu und bearbeiten Sie die bestehenden. In den Konferenzunterlagen finden Sie eine Kopie der hier gezeigten Info.plist. Einfach die Daten ändern.

Slide 13 – Ballastloswerden

FileMaker Runtimes sind schwergewichte und tragen noch viel PPC Code mit sich rum. Eine Richtlinie von Apple ist, dass kein PPC Code in der App enthalten sein darf. Das Werkzeug „XSlimmer“ leistet hier gute Dienste zum Entfernen von unnützen Code und Sprachen. Dazu gleich später.
Was Xslimmer nicht entfernen kann sind mindestens 2 Dateien, da diese reiner PPC Code sind: OleWrapper und OleWrappeRPCServer. Löschen Sie diese beiden Dateien. Die Runtime wird auch ohne diese Dateien funktionieren.

Slide 14 – Xslimmer

Nun kommt Xslimmer zum Einsatz. Benützen Sie Xslimmer je einmal für das Applescript.app und einmal für die Runtime.app.

Slide 15 – App zusammenbauen

Setzen Sie die App zusammen.
Info.plist
Applet.icns
Runtime.app mit allen notwendigen Dateien und der Datei FMApp.icns

---

Slide 16 – Berechtigungen

Mit Propedit setzen Sie die Berechtigungen, damit die Terminal Befehle richtig ausgeführt werden.

Slide 17 – App Signieren

Geben Sie nun im Terminal den Befehl für das Signieren ein. Achten Sie darauf, dass der Pfad zur App korrekt ist und auch der Zertifikatsname richtig ist.

Slide 18 – Installationspaket erstellen und signieren

Geben Sie nun im Terminal den Befehl ein. Achten Sie darauf, dass Sie das Zertifikat für den Installer angeben und nicht das Developer Zertifikat.

Slide 19 – Installation testen

Testen Sie das erstellte Installationspaket, indem Sie diesen Befehl eingeben. Und nun geht es ans testen des Apps.

Slide 20 – App hochladen

In iTunesConnect gehen Sie auf „Manage Apps“ und melden Sie das Hochladen der App an. (Rechts oben „Ready to Upload Binary“.

Das Hochladen selbst übernimmt der Application Loader (teil vom Xcode Installationspaketes).

Nun wird ein mehrstufiger Test für die App durchlaufen.
1. Stufe: Application Loader prüft das Installationspaket ob dieses über die Info.Plist mit den notwendigen Einträgen verfügt.
2. Stufe: iTunesConnect prüft sofort nach erfolgten Hochladen ob der Code gültig ist (PPC) und ob die App korrekt signiert wurde.
3. Stufe: Ein Mensch prüft innerhalb der nächsten Wochen! ob die App den Richtlinien entsprechen.
