



Kántor Tibor



“iOS 8 the most significant change for developers since the introduction of the original iPhone OS SDK.”



Tematika

Miről lesz szó?

- **Adaptive Layout** - képernyőméret független felület tervezése
- **Handoff** - Munkamenet folytatása
- **Live Rendering** - egyedi vezérők az Interface Builder-ben
- **App Extensions** - alkalmazás kiterjesztések
- **CloudKit** - Apple-féle szerver szolgáltatás
- **WebKit** - UIWebView helyett
- **Vizuális effektusok**

Miről nem lesz szó pl.?

- Programozás Swiftben
- Metal - Apple-féle grafikus API 2D és 3D rajzoláshoz az OpenGL helyett
- Adaptációról mélyebben- ViewControllererek az adaptív világban
- Custom ViewController prezentációk
- SceneKit - SpriteKit 3D-s megfelelője
- Photos Framework - Fotók hozzáférése

Mire lesz szükség?

- Xcode 6.1+
- Minimális Swift tudás
- Auto-Layout ismerete
- Apple watch 😊
- <http://goo.gl/kjAN0u>



Adaptive Layout

Történet

- Kezdetben: Spring And Struts
- iOS 6-tól Auto Layout
- iOS 7: Auto Layout finomítások
- iOS 8: Adaptive Layout: az Auto Layoutra építve

Adaptive Layout

- Mi indokolja az új technológiát?
- Ők:



- 5 különböző méret, ami valójában 10!
- ...és mi jön még?

Adaptive Layout

- Segítségével képernyő méret független felhasználói felületet lehet készíteni
- Alapja az AutoLayout
- Követelmény: Xcode 6
- Egyetlen Storyboard fájl fedi le az összes képernyőt
- A forgatást is ezzel kezeljük
- Visszafelé kompatibilis egészen iOS 6-ig

Adaptive Layout - Alapok

- Méret osztályok: minden fizikai eszköz (egy orientációjának) a vízszintes és függőleges mérete egy adott méretosztályba sorolható.
- Két méret: regular és compact
- Két fajta osztály: Vertical Size Class és Horizontal Size Class
- A designolás java része a méret osztályok figyelembe vétele nélkül is megtethető!

Adaptive Layout - Alapok

- Méret osztály mátrix:

	Vertical Size Class	Horizontal Size Class
iPad Portrait	Regular	Regular
iPad Landscape	Regular	Regular
iPhone Portrait	Regular	Compact
iPhone Landscape	Compact	Compact
iPhone 6 Plus Landscape	Compact	Regular

- Ezek bármikor felül definiálhatók kódból, ha szükség van rá

Adaptive Layout - Háttérben

- Új osztály: Trait Collection
- Megadja a felhasználói felület jellemzőit
- 4 property:
 - > horizontalSizeClass (Compact)
 - > verticalSizeClass (Regular)
 - > userInterfaceIdiom (Phone)
 - > displayScale (2.0)
- A TraitEnvironment protokollt megvalósító osztályok: UIScreen, UIWindow, UIViewController és UIView



Adaptive Layout – Alapok

- Az egyes értékek lehetnek unspecified is, ha nem számít
- Gyerekek öröklik a szülő TraitCollection-jét
- Felül definiálhatók
- Összeadhatók

Adaptive Layout – Forgatás

- A `UIInterfaceOrientation` és `UIUserInterfaceIdiom` deprecated
- Az adaptív világban: `TraitCollection` megváltozik erre lehet reagálni
- Új függvény: `func willTransitionToTraitCollection(_ newCollection: UITraitCollection, withTransitionCoordinator coordinator: UIViewControllerTransitionCoordinator)`

Adaptive Layout – Interface builder

- Universal Storyboard: egyetlen storyboard fájl, amely az összes méretosztály összes design-ját tartalmazza
- Új Assistant editor preview, amiben minden méretosztály megjeleníthető
- Az alap méretosztály az wAny hAny, lehet változtatni
- Az egyes méretosztályokra vonatkozó kényszerek külön telepíthetők és törölhetők

Adaptive Layout – Erőforrások

- Korábban normál és @2x képek
- Új @3x a retina HD (iPhone 6 plus) miatt
- De az Xcode 6-tól méret osztályok szerint is lehet különböző képeket megadni ezeket a rendszer automatikusan tölti be
- Alapból az w any h any, csak a pluszban felvett méretosztályok képeit tölti be
- Tipp: @3x-es megoldást csak korábbi projektek frissítéseseinél használjuk!

Adaptive Layout – Erőforrások

- A font méretek is megadhatók méretosztályok szerint,
 - > alpból az wAny hAny-t tölti be mindenhova
 - > attributedString-et nem támogatja
- View-k adaptívan instalálódnak
 - > Tulajdonságoknál az installed property

Adaptive Layout – folyamat

1. Tervezés és designolás wAny hAny méretben
2. Ha szükséges, akkor az egyes méretosztályokra új kényszerek megadása vagy meglévők törlése
3. Képek hozzáadása méretosztályok szerint
4. Tesztelés a Preview-val, debuggolás a View debuggerrel

Adaptive Layout gyakorlat

Kitérő: Presentation Controllers

- A felugró ablakok kezelése Presentation Controllerekkel történik, amelyek adaptívan alkalmazkodnak a környezethez
- Felügyelik a megjelenítést és animációkat
- Két közvetlen leszármazott
- UIAlertController: UIActionSheet és a UIAlertView-t fogja egybe
- UIPopoverPresentationController: UIPopoverController

Handoff áttekintés

Handoff

- Felhasználói munkamenet folytatása más eszközökön
- Vagyis a felhasználói aktivitás megosztása iOS-es eszközök, illetve iOS és OS X között
- Pl. Safari böngészés Mac-en és közeli iOS eszközön folytatása vagy

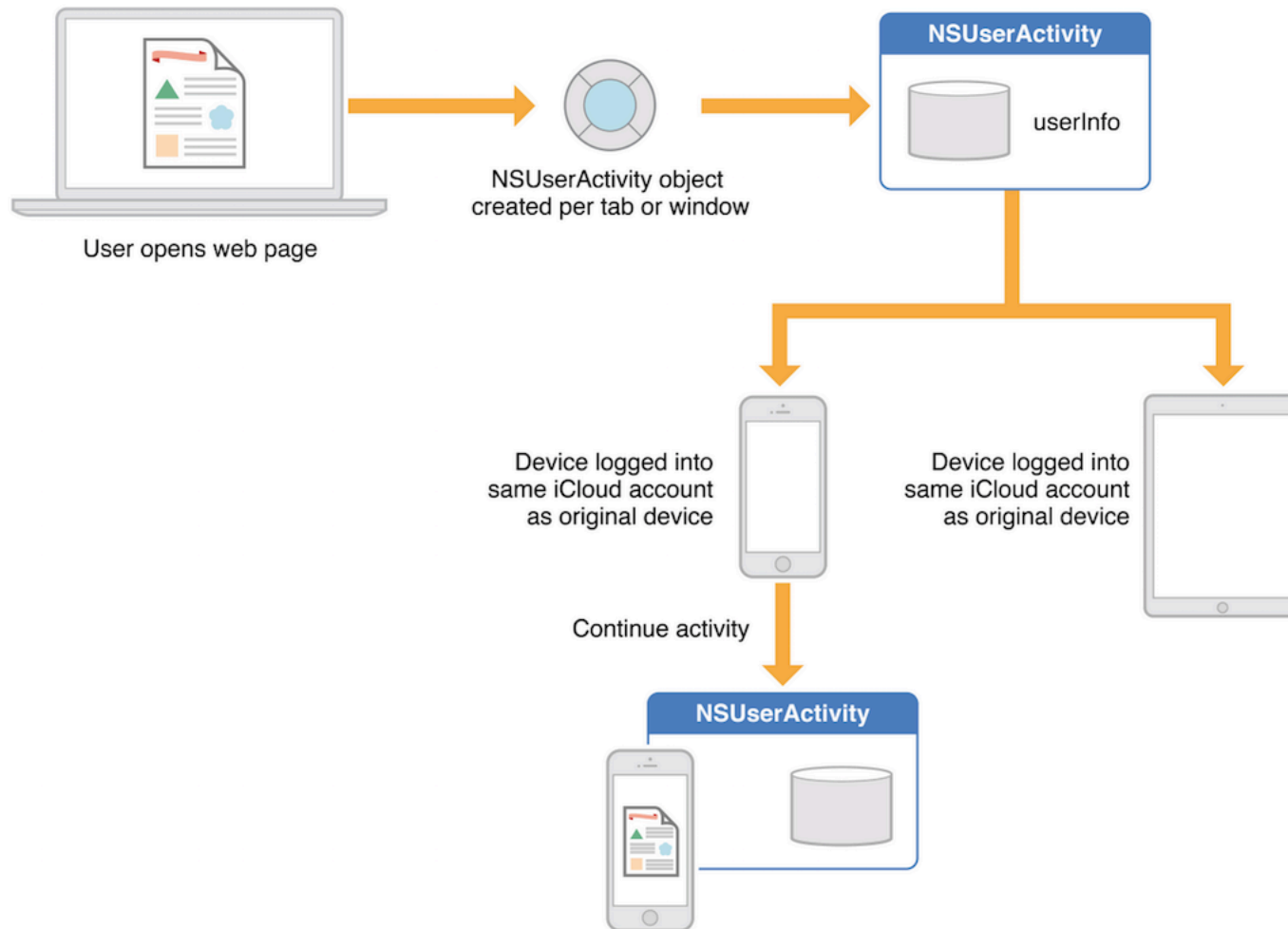
Handoff – mi kell hozzá?

- Ugyanaz a bejelentkezett iCloud account
- Bluetooth 4.0-ás eszköz (iPhone 4S +)
- Eszközök összerendelése
- Szimulátorban nem működik ☹
- Teszteléshez két eszköz kell

Handoff alapok

- Alapja: `UserActivity` osztály
- Ez az objektum utazik az eszközök között, ebbe kell becsomagolni az adatot, amely az állapot megállapításához kell
- `UserInfo` dictionary-be lehet pakolni (ami bármilyen `NSCoding`-ot megvalósító adat lehet)

Handoff alapok



Handoff – aktivitás

- Activity típus: reverse DNS, amely azonosítja az aktivitást, hasonlóan az URL sémákhoz
- Lehetséges különböző alkalmazások között is handoffot megvalósítani (gondoljunk csak a MacOS -> iOS handoff-ra)
- Megosztás feltételei:
 - > Ugyanaz a developer team
 - > Ugyanaz az activity típus
 - > Alárt alkalmazás

Handoff működés - küldés

1. Létrehozzuk a `UserActivity`-t:

```
let activity =
```

```
    NSUserActivity(activityType: "com.bme.autsoft.pelda")
```

```
    activity.title = "Viewing" activity.userInfo =
```

```
    ["shopsnap.item.key": ["Valami", "Másik", "Banana"]]
```

```
    self.userActivity = activity;
```

1. Beállítjuk aktuális `activity`-nek, ezzel indítjuk a frissítési folyamatot

2. `self.userActivity?.becomeCurrent()`

3. Ennek hatására az aktuális `ViewController`en hívódik meg a

4. `updateUserActivityState(activity: NSUserActivity) -> void`

Handoff működés - fogadás

1. A fogadó oldalon az AppDelegate-ben:
2. `application(:willContinueUserActivityWithType:),`
3. Majd, ha letöltötte az aktivitást:
4. `UIApplication!, continueUserActivity
userActivity: NSUserActivity!,
restorationHandler: (([AnyObject]!) ->
Void)!) -> Bool`

Live Rendering

Saját vezérlők tervezése

- Korábban saját vezérlőt csak és kizárólag a szimulátorban vagy eszközön lehetett tesztelni
- iOS 8 és Xcode 6-tól tesztelés és debuggolás közvetlenül az Interface Builder-ben
- Bármilyen UIView és annak leszármazottai

Használatának menete

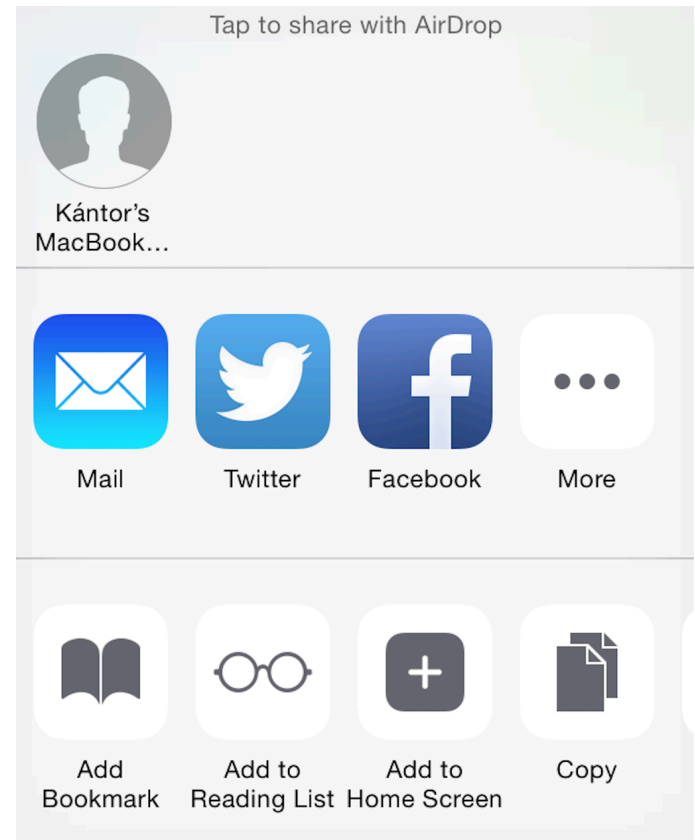
- Jelezni kell a fordítónak, hogy Interface Builder-ben megjelenítendő:
`@IBDesignable (IB_DESIGNABLE)`
kulcsszó az interface deklaráció fölé
- Hatására a megváltozik az identity inspector
- `prepareForInterfaceBuilder()` - csak design időben szükséges kódhoz
- `#if !TARGET_INTERFACE_BUILDER`

Használatának menete

- Propertyk kiajánlása az interface builder felé: `@IBInspectable` (IBINSPECTABLE):
- `@IBInspectable` var
- `variableName` Engedélyezett típusok: `Int`, `CGFloat`, `Double`, `String`, `Bool`, `CGRect`, `CGSize`, `CGPoint`, `UIImage` and `UIColor`
- Debugolás az interface builderben

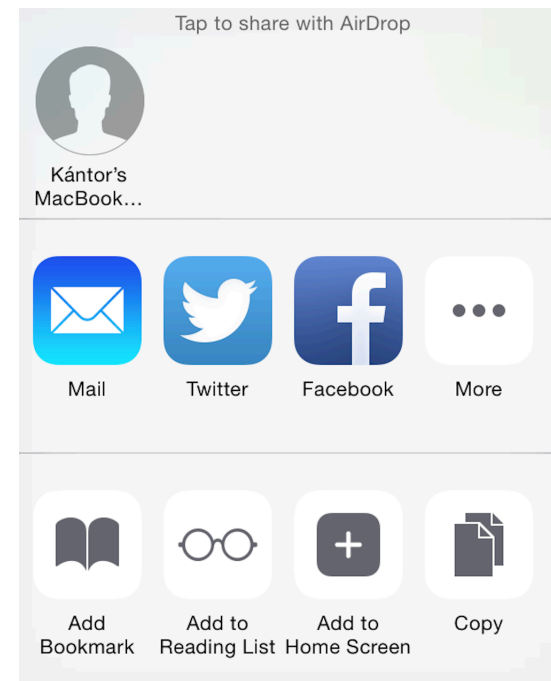
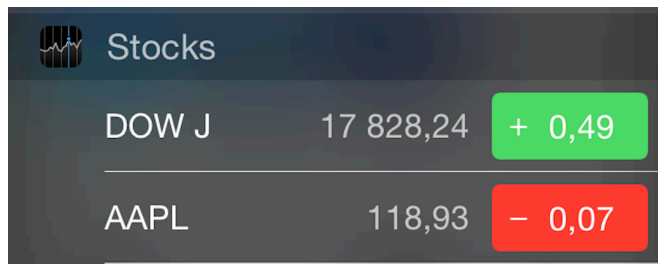
Live Rendering gyakorlat

App extensions



App extensions

- Alkalmazás kiterjesztések, amelyekkel más (akár a beépített alkalmazások) funkcionálisát lehet kibővíteni.
- 6+1 különböző típus



App extensions

- Today extension
 - > “A Widget.”
 - > A Notification Center Today nézetében jelennek meg.
 - > Apró ViewControllerek, amelyek valamilyen információt jelenítenek meg
 - > Pl: Stocks, Weather
- Custom Keyboard extension
 - > Third party billentyűzetek a beépített helyettesítésére
 - > Pl. Swype

Alkalmazás kiterjesztések

- Share Extension

- > Adat megosztás alkalmazások és egyéb szolgáltatások között
- > iOS5-től jelent meg a Tweeteléssel, fokozatosan bővült
- > A mi alkalmazásunkkal vagy más szolgáltatással, amely rendelkezik a megfelelő apival
- > Activity Controllerben, a felső sor pl: Facebook, Flickr, iCloud Photo Sharing

Alkalmazás kiterjesztések

- Action Extensions

- > Valamilyen adat megtekintése és áttanszformálása egy másik alkalmazáson belülről.
- > Activity Controller-ben az alja
- > Pl. 1Password hozzáférés a jelszóhoz bárhonnán, Assign to Contact, stb

- Photo Editing extentions

- > Az Apple Photos alkalmazásába beépülő mini alkalmazás, amely a képet módosíthatja
- > Tipikusan valamilyen speciális szűrő vagy effekt az alkalmazásra
- > Elérésük: Photos-on belül edig majd a három pont

Alkalmazás kiterjesztések

- Document Provider Extensions
 - > Document picker view controller lehetőség, amivel importálni, exportálni, mozgatni és megnyitni lehet külső alkalmazással
 - > Dokumentumok (és fájlok) elérhetővé tétele más alkalmazások számára,
 - > Pl. Pages open from

Alkalmazás kiterjesztések



- WatchKit extension
- Apple Watch early 2015...
- WatchKit Xcode 6.2-től.
- Valójában a Watch alkalmazás is egy Extension
- Három típusuk:
 - > Watch App: hibrid alkalmazás UI a Watch-on, logika a telefonban
 - > Glance: aktuális alkalmazás státusz megjelenítése az órán
 - > Notifications: customizálható értesítések megjelenítése

Fejlesztői szemmel 1.

- Nem önmagukban létező alkalmazások, hanem “csak” kiterjesztések
- Az alkalmazás bundle része, a kiterjesztendő alkalmazás a **konténer** App
- Minden egyes kiterjesztéshez saját API-k tartoznak
- Saját processzben futnak függetlenül a konténer app processzétől (akár többen is)

Fejlesztői szemmel 2.

- Saját binárisuk van amely független az alkalmazás binárisától
- A konténer appnak is csinálni kell valami értelmeset, különben reject
- Majdnem minden API elérhető, kivételek: UIKitUI, UIApplication
- Alapból nem látszanak, engedélyezni kell,
- Tilthatók és újrarendezhetők

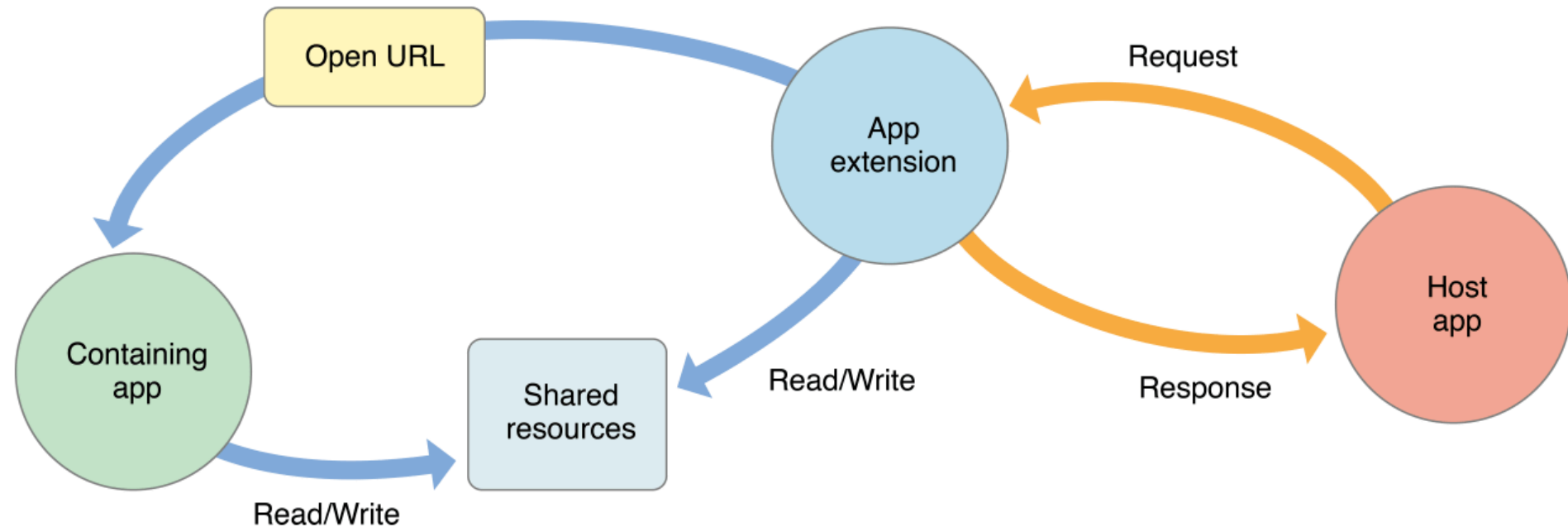
Működésük

- Az extensiont futtató alkalmazás a **host** app.
- Amíg fut kizárólag a host appal tud kommunikálni, nincs interprocess kommunikáció vagy hasonló
- A konténer alkalmazással sem tud direktben kommunikálni ☹

Kommunikációs lehetőségek

- Két lehetőség:
- Shared data container
 - > Olyan tároló ahová mindkét app írhat
 - > Valójában NSUserDefaults
 - > Kell hozzá App Groups capability
 - > Pl. ne kelljen feleslegesen újra letölteni
- Open URL
 - > Konténer app megnyitása

Összefoglalva



Működésük – Saját plist

- Saját info.plist fájl, benne az NSExtension dictionary-val (amely kiterjesztésenként eltérő), legfontosabb elemei:
- NSExtensionPointIdentifier - típus
- NSExtensionActivationRule - meghívásra vonatkozó szabályok
- NSExtensionMainStoryboard - melyik a storyboard fájl legyen a belépési pont

App extension gyakorlat

CloudKit áttekintés

CloutKit

- Apple ingyenes BAAS szolgáltatása
- Valójában csak Cloud Storage
- Csak Apple eszközökön érhető el (de OS X és iOS is!)
- Nem CoreData a Cloutban (nem magát a CoreData fájlt tároljuk fent, hanem a tényleges adatok (rekordok) tárolódnak fent, mint kulcs érték párok)!
- “Apple specifikus Parse”

Miért érdemes használni?

- “Ingyenes”
- Könnyű használni:
 - > rendelkezésre állnak az API-k,
 - > felhasználónak elég bejelentkezni az iCloudba, nem kell a szerverekkel foglalkozni
- Megbízhatóság: az Apple vállalja a biztonságot

Miért **nem** érdemes használni?

- Apple only
- Szerverben logika egyáltalán nincs
- Megbízhatóság: az Apple vállalja a biztonságot

CloudKit fejlesztői szempontból

- CloudKitDashboard
 - > Konfigurálás
 - > Kezdeti adatok feltöltése
 - > <https://icloud.developer.apple.com/dashboard>
- Új API készlet
 - > Új függvények és osztályok az adatok eléréséhez

ClouKit alapok – Konténer

- Az adatok un. konténerben tárolódnak
- Ezek megoszthatók alkalmazások között, de más fejlesztő konténeréhez nem lehet hozzáférni
- Alapból egy konténer, de lehet többet is csinálni
- Konténert nem lehet törölni.

ClouKit alapok – Konténer

- Minden konténer két adatbázist tartalmaz:
 - > Publikus: fejlesztő által előre beállított adatok, mindenki olvashatja, de csak a fejlesztő írhatja
 - > Privát: csak a felhasználó olvashatja és írhatja
- Adatbázis: CKDataBase
- Feltöltése: kódból vagy Dashboard-on
- Egy rekord: CKRecord példány

CloudKit – adatok

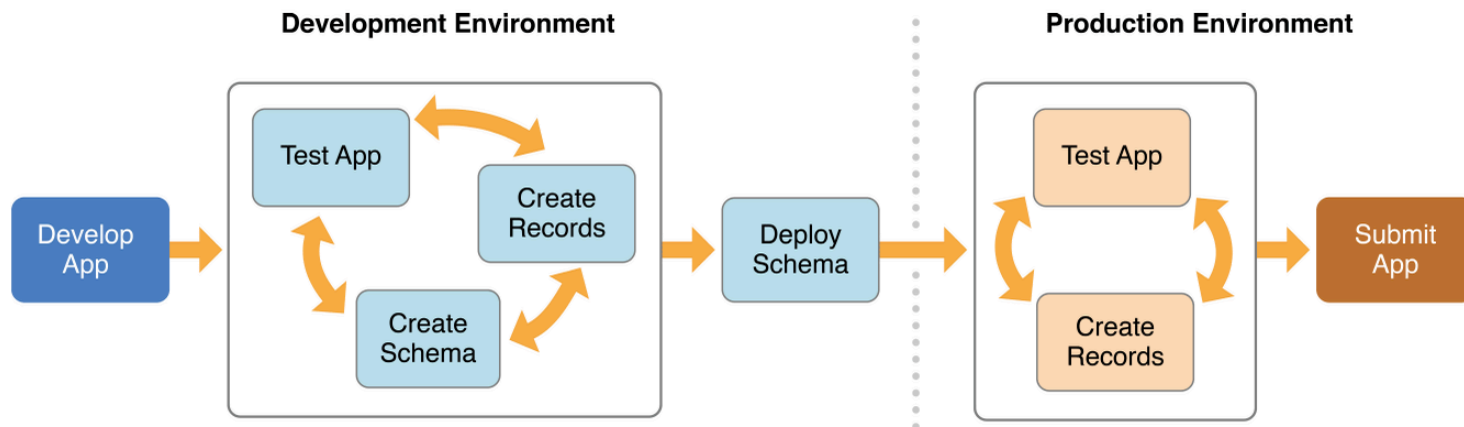
- Mit tárolhatunk egy CKRecordban?

Asset	nagy fájl
Byte	byte folyam
Dátum és idő	NSDate
Location	CLLocation
Reference	Kapcsolat
String	NSString
List	NSArray

- Használata hasonló a CoreData rekordokhoz (pl. ugyanúgy NSPredicate-tel szűrhetők stb.)

ClouKit alapok – Környezetek

- Két különböző környezet: development és production,
- Release után az utóbbiban csak a publikus DB adatai változtathatók (a séma sem)



WebKit áttekintés

WebKit

- A UIWebView nehézkes, lassú és még memória szivárgás is jellemzi
- Ellenben a Safari gyors és még a JavaScriptet is kellő sebességgel futtatja
- új Framework: **WebKit**, benne a WKWebView-val, amely felváltja a korábbi UIWebView-t
- Rendszerben több helyen elrejtve, pl. Messages

WebKit - Jellemzők

- Core Animation-re és hardveres gyorsításra építő scrollozás (60 fps!)
- Beépített gesztúra támogatás
- Nitro-ra (Safari) épülő Javascript motor
- A weboldal betöltésének teljes státusza
- Többfajta natív kommunikáció

WebKit – Natív kommunikáció

- Két új API-val:
- UserScripts: JavaScript kód beszúrása a webappba dinamikusan vagy betöltés kezdetekor vagy végekor, illetve, hogy mely framen történjen a megjelenítés, a tartalomba teljesen bele lelehet nyúlni (reklámok eltüntetésétől kezdve)
- ScriptMessages: JavaScript által generált üzenetek (NSNotification jellegű üzeneteket) generálnak, amelyek automatikusan sorosítódnak Swift (ObjC) objektumokra, feliratkozhatunk rájuk és szabad kéz

WebKit – Felépítés

- Egyetlen osztály és protokoll helyett 14 új osztály és 3 protokoll :)
- A WKWebkit interface nagyon hasonló a WebView-hoz, sok függvény ugyanaz, de persze a WebKitWebView sok újat tartalmaz

WebKit – Protokollok

- WKNavigationDelegate: navigáció és betöltés
- WKScriptMessageHandler: java script eseményekre való reagálás
- WKUIDelegate: natív UI elemek megjelenítésére használható

WebKit – Osztályok

- **WKBackForwardList**: Meglátogatott lapok
- **WKBackForwardListItem**: Előbbi egy eleme
- **WKFrameInfo**: Frame adatai
- **WKNavigation**: weboldal betöltések követése
- **WKNavigationAction**: Navigációt kiváltó események követése
- **WKNavigationResponse**: navigációs válasz (pl. jogosultságkezelésnél jön jól)
- **WKPreferences**: Beállítások
- **WKProcessPool**: Webes processek poolja
- **WKUserContentController**: Javascriptes kommunikációra
- **WKScriptMessage**: Ebbe csomagolódik a script által küldött üzenet
- **WKUserScript**: A injektálható javascript osztálya.
- **WKWebViewConfiguration**: Ezekkel inicializáljuk a webview-t
- **WKWindowFeatures**: Az ablakozási tulajdonságok új WebView kérésekor

Vizuális effektusok

Vizuális effektusok

- iOS 7: rendszerinten blurözött nézetek: navigationbar, tabbar, toolbar
- Saját appban: third party statikus megoldások vagy hackelés a toolbar-ból kiindulva
- iOS 8-ban új API
- A third party megoldásokkal szemben “live” vagyis nem statikus

Vizuális effektusok

- `UIVisualEffectView`: új konténer nézet
- Meglévő View-khoz lehet hozzáadni, mint alnézetet (subview)
- Minden `EffectView` tartalmaz egy effekt (`UIVisualEffect`) leszármazottat
- Két féle egymásra épülő effektus:
 - > **`UIBlurEffect`**: megszokott blurös effekt (3 féle színárnyalattal)
 - > **`UIVibrancyEffect`**: A blurözött viewn lévő tartalom “vibrációja”

Vizuális effektusok gyakorlat

Összefoglalás

- “iOS 8 the most significant change for developers since the introduction of the original iPhone OS SDK.”
- Tényleg sok az újdonság és lehetőség
- És ez még csak a kezdet...



Köszönöm a figyelmet!