# Java

* **JavaDoc**: A html sokkal értelmesebb választás javaDoc-hoz, mint egy külön xml definálása, mint C#-nál
* Apróság: **Java**-ban, ha **switch**-elünk egy enum-ot, akkor nem pl UserOrderStatus.Finished hanem csak Finished lesz a case –ekben. C# -ban nem

# Normal

* a **Gradle** Windows-on nem a legjobb. Sokszor kellett újraindítanom a gépet (néha elég volt csak az IDE-t), mert GradleConnectionException miatt nem tudott fordítani
* **GradleConnectionException**: ez egy kurva nagy faszság! Hányszor volt, hogy elindul a gépem nulláról, és rögtön nem tud fordítani, mert ez jön! Egyelőre csak az újraindítás segít…
* A **Fragments** –ekkel jobban darabolható a kód, mint WP-on
* WP-n a **silverlight** miatt amúgy is **sokmindent egybe** kellett írni. Pl talán ResourceDictionary se volt… Csomó mindent az app.xaml fájlba kellett rakni!
* Android: meglepően dinamukusan ment a **proxy**: volt és működött a beépített süti kezelés, anélkül, h fél napokat kellett vna kódolni hozzá!
  + Nem volt viszont WSDL-ből proxy-t generáló cucc. WP-nél legalább volt, még ha használhatatlan is, mert a süti kezelés silverlight-ban kb kivitelezhetetlen…
* Android-on sokkal több **UI dolgot kellett kódból írni**, mint WP-n. Viszont ez így könnyebb volt, persze hosszú távon melyik a jobb, az passz…
* WP volt később, **szebb az API**-ja is. Pl tud olyat, h a képnek src-ben URL-t állítunk be, és kész. Nem kell kézzel megírni a letöltő logikát
* Mindkettőben van **design time data**, bár WP-n lényegesen dinamikusabb a dolog
* Mindkettőre igaz: felejtsd el, ha valami nem-basic dolgot tudtál Java-ban/C#-ban, itt úgyse használhatod
  + Android: anonym Fragment –et tilos használni, mert nincs default ctor-ja, így a keretrendszer nem tudja össze-vissza törölgetni meg újra példányosítani kedvére.
* Android: ez a **folytonos Activity**/**Fragment** **törlés** elég nehézkessé teszi a backendről egyszer már letöltött adatok tárolását. Főleg, mert a Bunble objektum, ami jár hozzá adat tárolás céljából, nem enged Object-et átadni; szóval nem referencia szerint adható csak át a state
* A XAML-ös megközelítés mindenképp egy jóval érettebb gondolat. Android-on eléggé hiányzik az **adatkötés**… Meg a ViewModel-ek is!
* A XAML sokkal ügyesebb, mint ez az Android xml… Egységes, mindenhol ugyanúgy tudom szerkeszteni a child elementeket (nagyjából). Míg android-nál minden esetben egyéni a cucc…
* Ctx menü máshogy műxik. WP-n bármihez beállíthatom, amihez csak akarom. Android-on… talán ott is…
* WP-nél jobban szét lehetett szedni a dolgokat. Pl a BookBlock kilistázás sokkal egyszerűbben és logikusabban ment ott, mint Android-on
* Android: *Fragment: nem rakhatod bele a layout-ba, ha ki akarod cserélni. Így már csak kódból használhatod*
* Android sokkal sokkal **imperatívabb**… Könnyű elveszni a részletekben
* Szerintem WP8-nál jobban „**beindult**” a fejlesztés egy idő után. Ugyan sokkal meredekebb lett vna a betanulási idő, ha nem ismerem a WPF-et korábbról; viszont talán az Android fejlesztés mintha nem lenne olyan gyors azután se, h beletanult az ember
* Android-ban több az **ismétlődő kód**
* A *GridLayout* android-ban igen gyenge. Ugye a Grid a legnépszerűbbek között van xaml-ben az egyszerűsége és logikussága miatt; sajnos sztem android-on nem, mert elég limitált…
* Az **AppBar** különbségekről tudok majd írni
* **Bejelentkezéskor** kattinthat inkább a regisztráció gombra, ekkor átviszi a regisztrációs oldalra, és átviszi az esetleg beírt adatait is. Sikeres regisztráció után visszalép a regisztrációs oldalról, ÉS HA a login-ről mentünk oda, akkor onnan is. WP-n ezt lehet, h nehezebb megoldani, mint Android-on (én legalábbis elég körülményesen oldottam meg, azt látom :D)
* Android-on ha **debug** közben látunk egy elkapatlan **exception**-t, szerencse is kell hozzá, hogy StackTrace-t is láthassunk + Ezért írtam egy unhandled-exception handlert a BookteraApplication.java-ban, ahol mindet log-oljuk. Persze, ez asszem csak a UI szál, de nem biztos
* Android-on mintha egyszerűbb lett vna megcsinálni a **Profilom** **és szerkesztése** részen, hogy egyszer kérjük csak le a profil adatokat, maradjanak konzisztensek a cache-ek a szerver állapottal, és frissüljön a view a szerkesztés után. Bár persze lehet csak azért, mert WP-n túlbonyolítottam… minden esetre érdemes lesz majd megnézni
* Android-ban mennyi gányolás van… Pl most egy TextView-t azt tudsz **bold**, **italic** állítani xml-ből. **Underlined**-ra csak úgy, ha resource-ot használsz, és ott <u>…</u> közé rakod a string-et :D baromi átlátható. Ha pedig kódból kell… na az very nasty
* Android: sok a felesleges osztály. Pl ListViewFragmentBase – UserOrderFragmentBase – UserOrder… ezek xaml-ben sokkal letisztultabbak, kevesebb a szemét
* **Android**: úgy tűnik, szívás van a **nested listView** –val
* Az **adatkötés hiánya** ugye akkor okozza a (felesleges) kihívásokat Android-on, amikor **interakció** kell…
* A sendOrder-t megcsinálni 4+ óra volt android-on. Ugye, adat kötés nélkül…
* WP: az URI navigation és a Page –es felfogás lényegesen egyszerűbb, mint az Android-os Activity-sdi. Van utóbbinak valami előnye vajon az előbbi felett?