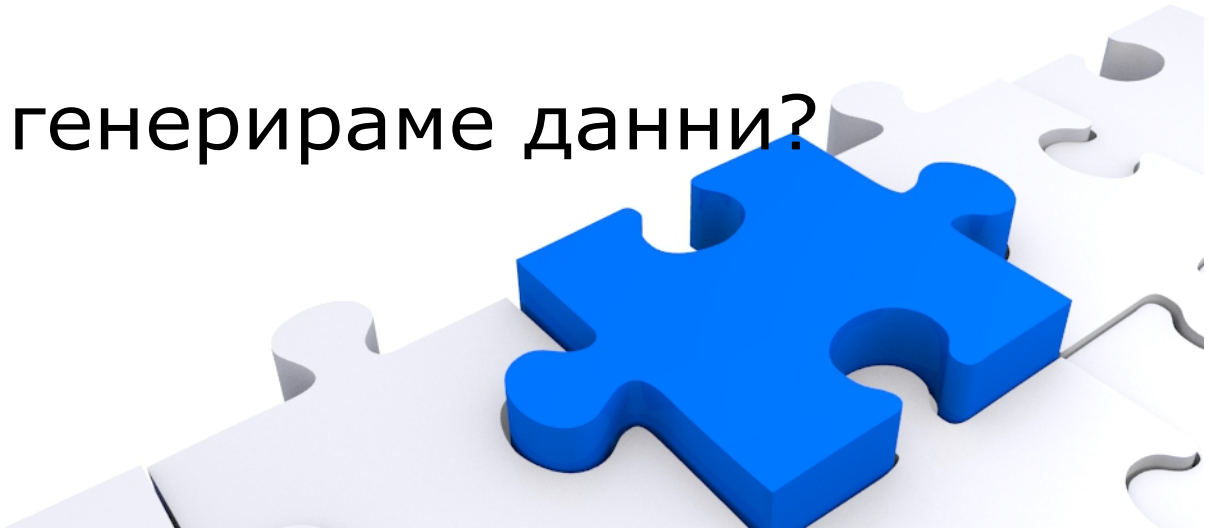
IIntent 2, more persistence, outside world

- Вместо съдържание:

- Intent(s) - formal;
- Още начини да си препълним паметта (more persistence);
- Само ние ли ще генерираме данни?



Intent Formal/1 – параметри и конструиране

- Или как да си построим Intent.
- От една страна Intent обекта носи информация по която се ориентира Android за да го адресира правилно; От друга – адресираният компонент, също има нужда от информация, така щото да отработи правилно „заявката“;
- Първична информация, необходима за Intent:
- Component Name, optional; Определя дали Intent-а е explicit или implicit;
- Ако сме го задали, това най-често е за компонент от нашето приложение, цитирайки пълният път (включително пакета) до него;
- Ако не сме – то тогава имаме implicit Intent; Системата решава, по други белези, кой компонент да „извика“; Белезите са действие (action), данни (data) и категория (category);
- Полето (filed) в Intent класа отговорно за това е ComponentName; Установяваме го с .setComponent(), .setClassName(), .setClass() или чрез конструктора на Intent класа;

Intent Formal/2 – параметри и конструуиране

- Action – String, указващ най-общо действието което трябва да бъде извършено;
- В случай на broadcast Intent, това е съобщението което се рапортува;
- Това е параметърът, който в най-голяма степен определя как се структуриран нашият Intent обект (тоест съдържанието на останалите 2 параметъра – data && extras);
- Можем да указваме и наши действия – такива които използваме в границите на нашето приложение, но обикновено използваме действия дефинирани от Intent класа (или други класове в Android);
- Примери:
- Intent.ACTION_VIEW, Intent.ACTION_SEND
- static final String ACTION_MYACT = "net.yaht.m.ACT01";
- Указваме действие със .setAction() или конструктора на Intent класа;
- Intent jdoc: http://developer.android.com/reference/android/content/Intent.html

Intent Formal/3 – параметри и конструуиране

- Data – Uri обект, указващ данните в/у които ще се въздейства; Освен това се използва за указване и на MIME типа на тези данни; Почти винаги зависи от избрания Action;
- Uri jdoc: http://developer.android.com/reference/android/net/Uri.html
- Специален тип uris, започващи с content: (или file:), указват че данните са налични в/у у-вото;
- За установяване на Uri обект - .setData();
- За MIME типа - .setType();
- За двете – setDataAndType();
- Конкретни пример – в документацията на Intent класа;

Intent Formal/4 – параметри и конструиране

- Category – Отново String, съдържащ допълнителна информация за типа компонент, който трябва да отработи този Intent; Съществуват достатъчно стандартни категории, но повечето Intent обекти не изискват категория;
- Пример:
- CATEGORY_BROWSABLE – указва че този компонент (най-често Activity) може да бъде изпълнен от browser за визуализиране на съдържание например (Можем да си направим собствен downloader например :));
- CATEGORY_LAUNCHER – този го знаем; Това входната точка на нашето приложение и се list-ва във системният application launcher;
- Четворката (component name, action, data, category) – са определящите характеристики на Intent; По тях Android определя какъв компонент да стартира (вкл. от кое приложение);

Intent Formal/5 – параметри и конструиране

- Extras – с други думи, всяко правило си има и изключение ;)
- Key-Value двойки, които носят допълнителна информация, необходима за това да бъде завършено съответното действие;
- Идеята на Uris – т.е. описват данни и биват извиквани от специфични ACTION(s)_*;
- Добавянето се осъществява с .putExtra(), имащ най-различни форми - според типа данни който добавяме (виж. Intet Jdoc);
- Освен това се поддържат и Bundle, http://developer.android.com/reference/android/os/Bundle.html обекти;
- Парадоксално, но „разбирането“ на Uri(s) и безбройните им типове, започва именно от изключенията за .putExtras, наследени от ранните вересии на Android (поддържани и до днес);
- Именно този начин използваме и при ACTION_SEND Intent-а в примера днес;

Intent Formal/6 – параметри и конструиране; Final words.

- Като основно свързващо звено в Android, темата за Intents е доста обширна; Зачеквайки нея е наложително да се обърне и внимание на куп други понятия; Заради това - summary на Intents (т.е. до къде сме стигнали);
- 3 основни начина на използване (use-cases):
- Стартиране на Acticity (със и без резултат, вътрешно и външно);
- Стартиране на Service (съвсем отделна тема, касаеща основно background/backstage task processing);
- Доставяне на broadcast; Съобщения, които всяко приложени може да получи; Тук са например системни (от Android) съобщения като „charging in progress“ или „system is booting“;
- Как да ги обработим, пак вид callback(s) на основа Activity Context, който можем да регистрираме динамично:
- http://developer.android.com/reference/android/content/BroadcastReceiver.html
- Освен това два основни типа – Explicit и Implicit;
- И четири параметъра с които определяме Intent - (component name, action, data, category);

Преди IIntent2, още малко Android Persistence

- Предишният път се запознахме със Internal Storage на Android и основни примитиви с които го достъпваме;
- Какво още ни предлага Android:
- Shared Preferences;
- External Storage;
- За любителите на структурирания подход към данните SQLite Databases;
- И разбира се мрежова свързаност;
- Днес ще се запознаем по-близо със Shared Prefs и External Stroage;

Преди IIntent2, още малко Android Persistence/2

- Shared Prefs – framework за примитивни типове данни; Основна характеристика – persistent, т.е. запазват се след restart, crash (или дори kill) на приложение;
- Можем да си запазваме всички примитивни типове данни, т.е. - float(s), int(s), long(s), и разбира се string(s).
- За какво ги използваме – основно настройки на приложението, запазване на състояние на Activity или просто спомагателни данни;
- Как ги използваме:
- Четене:
- prefs = getSharedPreferences(„<particular-prefs-name“, ACCESS_MODE);
- prefs.getBoolean(„my_bool_flag“, false); // 2рото е стойност по подразбиране;
- Писане:
- editor = prefs.editor();
- editor.putInt(„my_int_key“, (int));
- editor.commit();

Преди IIntent2, още малко Android Persistence/3

- Разбира се, на къде без tool-set за това да запълним и външната памет на даден Android – тоест картата му с памет;
- Ключово за External Storage:
- Искаме достъп за писане и/или за четене (т.е. обявяваме го в manifest файла на приложението):
- <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
- Проверяваме преди да използваме isExternalStorageWritable - в кода на IIntent2;
- Environment.getExternalStorageDirectory() - като път;
- Конструираме File обект, показващ пълният път до файла;
- От там насетне познатите ни FileInputStream/FileOutputStream на база този File обект;
- Още за Environment: /и това да достъпим всички предефинирани директории на картата ни с памет :)/
- http://developer.android.com/reference/android/os/Environment.html

Преди IIntent2, още малко Android Persistence/4; Final.

- Прекрасно summary за всички native storage възможности на Android:

  http://developer.android.com/guide/topics/data/data-storage.html
- Една бележка – handle with care ;)
- Кой точно option ще използваме, зависи от конкретният use-case на нашето приложени и /или activity;
- Shared Preds, Internal Cache са удобни за малки по обем данни;
- Масовият storage – external е по дефиниция бавен;
- SQLite db – е супер, но за много мънички по обем релации (няколко десетки реда); Иначе се превръща във performance bottleneck;

Примерът днес – IIntent2;

- Основна идея – с минимални усилия и разчитайки основно на Implicit intents да реализираме макс. Функционалност ;)
- Едно Activity по подразбиране;
- Действие A - Implicit intent за Android Camera App – тоест снимаме снимки, „искаме“ ги от App-а и си ги помним;
- Действие B - Генерираме си собствени картинки; Помним и организираме и тях; С други думи, колко трудно се ползва Bitmap/Canvas/Paint (image processing!);
- Действие C – отново Implicit intent, но този път за Android Email App; Тоест това което генерираме/снимаме да изпратим като e-mail (+ attachment разбира се);
-

IIntent2 – extra furher info;

- Прекрасен Camera turorial от Google:

  http://developer.android.com/guide/topics/media/camera.html

- Включително с насоки, как да си напишем сами Camera App (доста system-layer изложение);
- Споменатите по горе класове за „рисуване“ в Android:

  http://developer.android.com/reference/android/graphics/Canvas.html

  http://developer.android.com/reference/android/graphics/Paint.html

  http://developer.android.com/reference/android/graphics/Bitmap.html

- И разбира се какво има на готово, що се отнася до media like файлове и това да ги управляваме:

  http://developer.android.com/reference/android/provider/MediaStore.html